Trabajo de Fin de Grado

Aplicación para la monitorización y defensa de sistemas Linux

AUTOR: Joel Barra Muñoz
TUTOR: Sergio Pastrana Portillo

Universidad Carlos III de Madrid
Escuela Politécnica Superior
Grado en Ingeniería Informática

Agradecimientos

He recibido el apoyo de muchas personas a lo largo del desarrollo del presente trabajo de fin de grado, pero sobre todo quiero agradecer la colaboración por su gran interés y apoyo, a mi tutor Sergio Pastrana Portillo, profesor de la Universidad Carlos III de Madrid.

Por otra parte, también quiero agradecer a mi familia, pareja y amigos, por su incansable apoyo tanto a lo largo de este proyecto, como en el resto de la carrera.

Resumen

El presente documento recoge el procedimiento seguido para la realización del Trabajo de Fin de Grado del alumno Joel Barra Muñoz, llevado a cabo bajo la orientación y supervisión del profesor Sergio Pastrana Portillo. Este documento supone el último requisito académico para la obtención del título de Graduado en Ingeniería Informática en la Universidad Carlos III de Madrid.

Esta sección muestra, de forma sintetizada, tanto una breve introducción a la problemática tratada en este trabajo, la cual será ampliada en el primer capítulo, como un recorrido por el contenido de todo el documento, destacando aquellas partes más relevantes del mismo.

Hoy en día, los sistemas informáticos se han convertido en una herramienta imprescindible en casi cualquier ámbito de la sociedad. Estos sistemas se encuentran presentes en nuestra vida cotidiana, incluso más de lo que un usuario común se puede llegar a imaginar. En tareas tan comunes, actualmente, como la de acceder a la web de una empresa, pedir cita en el médico a través de Internet, consultar los puntos del carnet de conducir, etc., intervienen una cantidad indeterminada de distintos sistemas informáticos con diversos grados de complejidad. Al fin y al cabo, desde el punto de vista de este usuario, el cual se encuentra en su domicilio, él sólo interactúa con su ordenador personal y, cómo mucho, con su router el cual permite que disponga de acceso a Internet.

Sin embargo, en cualquier tarea similar a las mencionadas, intervienen otros sistemas informáticos, los cuales resultan totalmente trasparentes para el usuario. Esto resulta posible debido, entre otras cosas, al desarrollo de protocolos que permiten el intercambio de información entre computadores, de manera que el usuario final es totalmente ajeno a las operaciones de bajo nivel que se llevan a cabo durante este proceso.

Además, muchos de los tipos de tareas mencionadas suponen que en los sistemas informáticos, pertenecientes a la entidad con la que el usuario está interactuando, se encuentren almacenados datos de carácter personal, los cuales requieren de medidas de seguridad para garantizar su confidencialidad, integridad y disponibilidad, acorde a lo estipulado en la Ley Orgánica de Protección de Datos (LOPD) [10]. Más aún, estas entidades dispondrán de sistemas informáticos, los cuales, por supuesto, también almacenan y trabajan con datos cuya propiedad pertenece a la entidad, y además requieren disponer de acceso a Internet. Estos últimos datos sólo deberían ser manejados por personas que trabajen en dicha entidad.

Puesto que los sistemas informáticos se han convertido en una herramienta con tanta importancia en nuestra sociedad, resulta imprescindible proteger los mismos. Sin embargo, ¿qué es exactamente lo que hay que proteger?¿El sistema físico, es decir, el hardware?¿Los programas que permiten gestionar todos los componentes del computador, es decir, el software?¿Los datos con los que trabaja el sistema informático? La mejor

opción es la de proteger el sistema en todos los niveles expuestos anteriormente: tanto a nivel de hardware, como a nivel de software y a nivel de datos con los que trabaja el software, los cuales son conocidos como datos de nivel de aplicación.

El principal motivo de tener que proteger los sistemas informáticos, junto al de las catástrofes naturales, es que existen usuarios que buscan obtener beneficios a través de lograr acceder, modificar o hacer indisponibles los recursos con los que trabajan los sistemas informáticos de ciertas entidades.

Para llevar a cabo esta protección, se usan mecanismos de seguridad con distintos fines: proteger los equipos de incendios o inundaciones, garantizar que sólo ciertos usuarios tienen acceso al sistema, garantizar que sólo ciertos usuarios tienen acceso a (o permiso de modificación de) los datos con los que trabaja el sistema, etc. Estos mecanismos pueden ser clasificados en función del momento en el que actúan:

- Mecanismos de prevención. Estos mecanismos tratan de minimizar la probabilidad de la ocurrencia de una violación del sistema de seguridad.
- Mecanismos de detección. Estos mecanismos tratan de minimizar la propagación de una violación del sistema de seguridad.
- Mecanismos de corrección. Estos mecanismos tratan de minimizar el daño de una violación del sistema de seguridad.
- Mecanismos de recuperación. Estos mecanismos tratan de minimizar las consecuencias de una violación del sistema de seguridad.

Nuestro trabajo se centrará en aquellos mecanismos de seguridad, enfocados tanto a la protección del software como a la de los datos, del sistema informático que permiten prevenir o detectar actividades perjudiciales que un atacante esté llevando a cabo en el mismo, es decir, se centrará en mecanismos de prevención y detección. Estos serán aplicados a través de cortafuegos y sistemas de detección de intrusiones. También hay que tener presente que este trabajo está enfocado a herramientas de seguridad, que implementan los mecanismos mencionados, disponibles para sistemas Linux.

Sin embargo, en muchos casos, por no decir en todos, el manejo de las herramientas que implementan esos mecanismos de seguridad no resulta trivial, principalmente, por dos motivos:

1. Es necesario un conocimiento básico acerca del funcionamiento de sistemas informáticos, como puede ser tener nociones del funcionamiento de redes de computadores, por ejemplo.
2. Son necesarios conocimientos acerca de cómo utilizar cada herramienta, cómo configurar la misma, qué valores pueden ser asignados a cada parámetro de configuración, etc.

Es en este punto en el que encontramos la motivación de nuestro trabajo: pese a que los obstáculos planteados por el primer motivo son prácticamente insalvables, debido a

que los conocimientos básicos requeridos para saber qué es lo que se está haciendo son indispensables, sí que es posible salvar los obstáculos planteados por el segundo de los motivos. La dificultad en el uso de estas herramientas se debe a que su configuración se suele llevar a cabo a través de comandos de consola, scripts de configuración y ficheros de configuración. Estos dos últimos suelen ser bastante extensos.

Por tanto, para facilitar la labor del administrador de seguridad de sistemas Linux, se tratará, a lo largo de este trabajo, de desarrollar una herramienta que permita realizar la gestión de algunas herramientas de seguridad de estos sistemas de una manera sencilla, cómoda y eficaz. De esta forma, la herramienta permite que este administrador pueda centrarse en las tareas que debe llevar a cabo, no teniéndose que preocupar por la manera en la que realizar las mismas.

Los mecanismos de seguridad que en este trabajo se han estudiado son:

- Los sistemas cortafuegos permiten llevar a cabo un control de acceso a nivel de la capa de red. De esta manera, cuando un paquete de red llega al equipo en el que se encuentra el cortafuegos, éste es capaz de decidir, en función de una política establecida por el administrador de seguridad, qué acción aplicar sobre el mismo. Estas acciones son, básicamente, las siguientes: permitir que el paquete de red continúe hacia su destino, descartar el paquete de red o recoger la llegada del paquete de red en un registro (log) del sistema operativo. La herramienta estudiada que aplica este mecanismo es IPtables, el cual es un cortafuegos que se encuentra preinstalado en la mayoría de las distribuciones Linux y que es ampliamente utilizado tanto en escenarios académicos como empresariales.

- Los sistemas de detección de intrusiones (IDS, Intrusion Detection System) permiten, a través del análisis del tráfico de paquetes de red, llamadas de sistema, registros (logs) de aplicaciones, etc., alertar al administrador de seguridad de la existencia de algún agente o elemento en el sistema que pudiera suponer una intrusión, entendiéndose como tal cualquier intento de comprometer la seguridad del sistema. Este análisis se puede realizar a nivel de host o a nivel de red, diferenciándose ambos tipos de análisis en el ámbito en el que es aplicado el mismo. Además existen distintas formas de llevar a cabo la detección, se puede tratar de una detección basada en firmas o una detección basada en anomalías. La herramienta estudiada que aplica este mecanismo es Snort, un popular IDS de libre distribución para sistemas Linux.

- Los sistemas de prevención de intrusiones (IPS, Intrusion Prevention System) permiten, además de alertar al administrador de seguridad de la existencia de algún agente o elemento en el sistema que probablemente sea un intruso, de tomar algunas medidas necesarias para expulsar a este agente o restringir las posibles acciones que puede llevar a cabo el mismo. Estos sistemas son muy similares a los presentados antes, incluso se puede considerar que son exactamente el mismo tipo de sistemas, pero con capacidad de reacción ante una intrusión. La herramienta estudiada que implementa este mecanismo es Snort_inline.

- Los sistemas de control de integridad permiten llevar a cabo una comprobación de qué ficheros han sido alterados, aún cuando su modificación no debía haberse llevado a cabo. Estos sistemas suelen aplicarse sobre ficheros críticos del sistema, como pueden ser ficheros de almacenamiento de contraseñas, ficheros de configuración del sistema operativo, etc. La herramienta estudiada que aplica este mecanismo es Advanced Intrusion Detection Environment (AIDE).

Tras un estudio realizado sobre los sistemas descritos anteriormente, se tomó la decisión de implementar en la aplicación la gestión y configuración del sistema de detección de intrusiones Snort, así como la definición de reglas del cortafuegos IPtables.

Existen en el mercado otras aplicaciones que permiten gestionar estas dos herramientas. Para la gestión de IPtables podemos encontrar las aplicaciones phpIPtables y PHP Firewall Generator, mientras que para la gestión de Snort podemos encontrar las aplicaciones Analisys Console for Intrusion Database (ACID), Basic Analisys and Security Engine (BASE) y SNEZ. Si bien todas ellas dejan ver algunos puntos de mejora, los cuales podrían ser incluidos en nuestra aplicación con el objetivo de situar ésta por delante de sus competidores más directos del mercado.

En cuanto a los aspectos más importantes recogidos en los requisitos de la aplicación, son destacables los siguientes:

- Se tratará de una aplicación web, por las ventajas que suponen este tipo de aplicaciones, ya que permite acceder a la aplicación desde cualquier computador con acceso a Internet y no requiere que el computador en el que se encuentra instalada la aplicación cuente con un entorno gráfico.

- La aplicación debe llevar a cabo la gestión de dos herramientas de seguridad (IPtables y Snort) cuya finalidad, como ya hemos mencionado, es completamente distinta.

- La gestión de Snort se llevará a cabo a través de la posibilidad de configuración de algunos de sus preprocesadores y algunas de sus variables de configuración.

- Será posible la visualización de las reglas de detección que Snort está aplicando.

- Será posible visualizar las alertas generadas por Snort a través de la herramienta BASE. Esta herramienta permite visualizar todas las alertas, permite la creación de gráficas, proporciona estadísticas de las alertas, etc.

- La gestión de IPtables se llevará a cabo mediante la configuración de todas las tablas que ofrece dicha herramienta. Esto incluye la gestión de las distintas cadenas de cada tabla y de las reglas de cada una de estas cadenas.

Por otra parte, para la implementación de esta aplicación, se encuentran disponibles una gran variedad de tecnologías que ofrecen distintas ventajas. Por lo tanto es necesario llevar a cabo un estudio de varias de estas tecnologías con el fin de identificar dichas ventajas y poder decidir qué tecnología será utilizada. En el caso particular de este

trabajo, se estudian las las tecnologías de Java Enterprise Edition (conocida como JEE), PHP Hypertext Pre-processor (conocida como PHP) y el marco de trabajo Grails. Entre los aspectos considerados a la hora de elegir una de ellas, se ha considerado:

- La experiencia que se tiene en el desarrollo de aplicaciones con cada una de estas tecnologías. Puesto que no se ha trabajado el mismo tiempo con estas tecnologías, el tiempo de aprendizaje de cada una ellas supone un factor muy importante a la hora de elegir con cuál se implementará la aplicación, ya que será necesario invertir más o menos tiempo tanto en aprender la sintaxis como la manera de trabajar de la tecnología.

- Si la tecnología permite desarrollar una programación modular. Llevar a cabo una programación modular y bien estructurada, es imprescindible para que posteriormente tanto el mantenimiento como la actualización de la aplicación se puedan llevar a cabo de una manera sencilla.

- Si la tecnología es multiplataforma, es decir, si la aplicación desarrollada con la tecnología podría ser ejecutada independientemente del entorno en el que esto sea llevado a cabo.

- Si existe un entorno de desarrollo para cada tecnología. Esto es de gran utilidad debido a que los entornos de desarrollo suelen automatizar tareas repetitivas, como por ejemplo la creación de clases, de métodos, constructores, etc.

Tras el estudio de todas estas características y haber valorado todas las tecnologías, se considera que la más apropiada para llevar a cabo este trabajo es JEE.

Recogida la información relativa a los requisitos de funcionalidad de la aplicación, y a cómo debe llevar a cabo su trabajo, se pasa a realizar el diseño de la misma. Como hemos dicho antes, el hecho de que este diseño se desarrolle de forma modular, es conveniente para que tanto la implementación, como el mantenimiento y actualización de la aplicación se lleve a cabo de manera sencilla. Esto se debe a que cada módulo en los que se dividirá la aplicación debe ser independiente del resto, por lo que los cambios en uno de ellos no afecten a la funcionalidad proporcionada por los demás. Este diseño modular permite que en un futuro se incorporen nuevas funcionalidades para la configuración de otras herramientas de seguridad, lo que es necesario en un mundo en el que la tecnología evoluciona a un ritmo vertiginoso. Para cumplir este objetivo, se recurre a una de las arquitecturas de software más conocidas a la hora de desarrollar aplicaciones web, la arquitectura Modelo-Vista-Controlador (MVC, Model-View-Controller). Ésta es una arquitectura en cuatro capas, cada una de las cuales lleva a cabo una labor bien definida en el funcionamiento de la aplicación:

- La capa Modelo se encarga de mantener la estructura de datos necesaria para que la aplicación desarrolle su actividad correctamente.

- La capa Vista se encarga de presentar al usuario la información recogida a través del modelo de datos, así como de ofrecer al mismo las posibles acciones que puede llevar a cabo.

- La capa Controlador se encarga de recibir las peticiones realizadas por el usuario, elegir qué acción aplicar en función de la petición y los datos recibidos, y finalmente, tras haber recibido la respuesta de la acción aplicada, decidir cuál es la siguiente vista que será mostrada al usuario.

- La capa Servicios se encarga de almacenar todas las posibles acciones que se pueden llevar a cabo en el sistema.

Además, dentro de estas cuatro capas de la arquitectura de la aplicación, encontraremos distintos módulos que contendrán todos los componentes necesarios para cumplir con cierta funcionalidad del sistema. Por ejemplo, dentro de la capa Servicios, se encuentran otros tres módulos:

- Un módulo para contener los componentes relacionados con la funcionalidad necesaria para el manejo de la herramienta Snort.

- Un módulo para contener los componentes relacionados con la funcionalidad necesaria para el manejo de la herramienta IPtables.

- Un módulo para contener los componentes relacionados con la configuración de la propia aplicación.

Un punto crítico de la aplicación es el diseño de la interfaz de usuario con la que contará la misma. Esto se debe a que es este elemento lo que el usuario percibe de la aplicación y a través de él es con lo que puede interactuar con la misma. Por tanto, los dos aspectos más importantes que se debe intentar garantizar son: que el usuario sienta que la aplicación se maneja de forma sencilla, y que el usuario no se vea obligado a tener que memorizar cómo llevar a cabo cada tarea (al fin y al cabo ésta es una de las desventajas que suponía configurar las herramientas de seguridad a través de la consola del sistema operativo o de los ficheros de configuración y que queríamos evitar). Para llevar a cabo un diseño que cumpla con estas características, es necesario tener claro tres cuestiones:

1. Qué cualidades tienen los futuros usuarios de la aplicación. De este modo es posible realizar el diseño de la interfaz de usuario teniendo en mente si estos usuarios cuentan con un perfil avanzado en cuanto al uso de cierto tipo de software, si es necesario incluir elementos de ayuda acerca de cómo navegar por la aplicación, etc.

2. Qué tareas pueden realizar los usuarios a través de la aplicación. De este modo es posible realizar el diseño de la interfaz de usuario teniendo en mente las tareas que los usuarios van a tener que llevar a cabo, siendo posible plantear la estructura de todos los elementos que aparecerán en dicha interfaz de un modo u otro.

3. Qué características debe cumplir la interfaz de usuario de la aplicación. De este modo es posible realizar el diseño de la interfaz de usuario teniendo en mente cuáles son todas las características que ésta debe cumplir.

Además de identificar estas tres cuestiones a lo largo del documento, también se toman ideas y conceptos de las interfaces de usuario de otras aplicaciones web que permiten la gestión de otros elementos software. En nuestro caso hemos tomado ideas de consolas de administración de servidores web y de la aplicación BASE. La ventaja que ofrece esto, es que el usuario final puede estar familiarizado con estas aplicaciones, por lo que si nuestra aplicación ofrece una apariencia y forma de trabajar similar a aquellas, la curva de aprendizaje del usuario será menos pronunciada.

Durante la fase de implementación resulta aconsejable llevar a cabo una serie de pruebas que permitan comprobar que el comportamiento de los componentes de la aplicación es el apropiado. Para que este proceso de pruebas se pueda realizar siguiendo cierto orden y evitando dejar componentes sin comprobar, en este documento se especifica el plan de pruebas que durante la implementación de la aplicación será llevado a cabo. En este plan se recogen los siguientes niveles de pruebas:

- Pruebas unitarias. Este nivel de pruebas permite comprobar el funcionamiento de los componentes de la aplicación a nivel interno de los mismos, siendo posible verificar si el funcionamiento de cada parte de estos es correcto.

- Pruebas de integración. Este nivel de pruebas permite comprobar el funcionamiento del conjunto de varios componentes de la aplicación.

- Pruebas de sistema. Este nivel de pruebas permite comprobar el funcionamiento de la aplicación completa.

Para cada uno de estos niveles se especifica un conjunto de pruebas, indicando en cada caso una serie de atributos que permitirán establecer, principalmente, en qué consiste la prueba, qué entrada hay que proporcionar a la prueba, qué salida debe producirse y qué condiciones (tanto antes de realizarse la prueba como después de realizarse la misma) deben cumplirse. En caso de que alguna de estas pruebas no cumplieran todos los requisitos para darse como satisfechas resultaría necesario llevar a cabo cambios en la implementación de la aplicación para que el funcionamiento del elemento probado sea el correcto.

Tras la conclusión de la fase de implementación de la aplicación, y antes de dar por terminado el desarrollo, se lleva a cabo una evaluación del producto obtenido. Antes de llevar a cabo esta evaluación resulta necesario especificar la manera en la que se hará la misma. En este trabajo, se ha establecido que la evaluación consistirá en completar una serie de tareas sencillas, las cuales serán llevadas a cabo por distintos usuarios (con capacidades que correspondan con las especificadas para los futuros usuarios de la aplicación). De esta forma, es posible obtener información para saber si se han cumplido los objetivos propuestos para nuestro trabajo. Además, se pueden obtener algunas ideas de cara a posibles actualizaciones de la aplicación, como ha ocurrido en nuestro caso.

Es necesario mencionar que durante el proceso de desarrollo aparecieron distintos problemas. Los más destacados, y los que mayor impacto han supuesto, han sido:

1. Durante la realización del plan de trabajo para el desarrollo de la aplicación, se sobrestimó el tiempo que sería posible dedicar a este trabajo durante el transcurso

del curso. En este plan de trabajo se recogían jornadas de cinco horas de trabajo diario, lo cual no fue posible llevar a cabo durante el curso, puesto que se dio prioridad al resto de asignaturas.

2. Algunos aspectos de programación a los que ha sido necesario enfrentarse nunca habían sido abordados, por lo que fue necesario dedicar algún día más de lo previsto a implementar esta funcionalidad de la aplicación.

Esto supuso tener que aumentar en tres meses más de lo planificado el desarrollo de la aplicación.

Finalmente, y tras haber completado todas las tareas necesarias para el desarrollo de la aplicación, podemos concluir que este trabajo cumple con todos los objetivos propuestos al inicio del mismo, obteniendo como resultado una aplicación sencilla e intuitiva que facilitará las labores de gestión de herramientas de seguridad sobre sistemas Linux. Además, puesto que la aplicación puede ser mejorada en algunos puntos, se proponen líneas futuras para actualizaciones que completen aún más la funcionalidad de ésta, como por ejemplo, por mencionar sólo alguna de ellas:

- Incluir la funcionalidad necesaria para que sea posible gestionar las reglas de detección de la herramienta Snort.
- Incluir la funcionalidad necesaria para que sea posible monitorizar la actividad generada por la herramienta IPtables.
- Incluir la funcionalidad necesaria para que sea posible gestionar y monitorizar la actividad de otras herramientas de seguridad.
- Incluir la funcionalidad necesaria para que sea posible llevar a cabo un control de acceso en la herramienta.
- Incluir la funcionalidad necesaria para que sea posible configurar los parámetros de la aplicación a través de la interfaz de usuario de la misma.
- Realizar un estudio sobre la simplicidad y usabilidad de la interfaz de usuario de la aplicación.

Abstract

This document describes the methodology and processes performed by student Joel Barra Muñoz during its bachelor thesis. This document represents the final academic requirement for obtaining the graduate in Computer Engineering by University Carlos III of Madrid. The thesis has been conducted under the guidance and supervision of Sergio Pastrana Portillo, teacher assistant in the Computer Science Department in the mentioned university.

This section firstly presents a brief introduction to present the objectives and motivations of this work, which will be further expanded in the first chapter. It also summarizes the contents of the entire document, highlighting the more relevant parts explained in each section.

Nowadays, computer systems have become an indispensable tool in almost every area of society. These systems are present in our daily lives, even more than any common user can imagine. In very common tasks, currently, as the access to any website, making an appointment at the doctor using the Internet, check the status of our driver license, etc., it should be used an undetermined amount of different computer systems with several degrees of complexity. Moreover, from the point of view of the user staying at his home, he only interacts with its personal computer and at most, with the router that allows the Internet access.

However, in any task similar to those mentioned above, there are many other Information Technology (IT) systems involved, which are totally transparent to the user. This is possible due to, among other things, the proliferation and development of communication protocols that allow the exchange of information between computers, being the end user unaware of the low-level operations that are performed during this process.

In addition, many of the tasks mentioned involve the use, storage and and transfer of personal data. This data, as stated in Spanish law "Ley Organica de Protección de Datos (LOPD" [10], require security measures to ensure its confidentiality, integrity and availability. Moreover, the entities using computer systems, also store and use data whose property belongs to the entity and which is a fundamental part of the economical actives of the company. Therefore this enterprise data must be handled by people working in that entity and with some restrictions and security measures.

As computer systems have become such an important tool in our society, it is critical to protect them. But, what exactly needs to be protected? Is it the physical system, namely, the hardware? Or are maybe the programs that allow users to manage the components of the computer, namely, drivers and operative systems? Or moreover, should be the security measures be applied to the data used in users applications? The best and only option is to protect the system at all levels discussed above, i.e., the data managed in the hardware level, the software level and the application level.

The main reason to protect computer systems, along with the possibility of having natural disasters, is that there are malicious users who seek to gain access, modify or deactivate the resources and information of the computer systems in certain entities.

To accomplish this protection, security mechanisms are used with different goals, as for example protecting the equipment from fires or floods, ensure that only certain users are authorized to access the system, ensure that only certain users have access to (or are allowed to modify) data used in the systems, etc.

These mechanisms are classified according to the moment in which they are applied:

- Prevention mechanisms. These mechanisms seek to minimize the probability of occurrence of a violation on the security system.
- Detection mechanisms. These mechanisms try to minimize the propagation of a violation on the security system.
- Correction mechanisms. These mechanisms try to minimize the damage of a violation on the security system.
- Recovery mechanisms. These mechanisms try to minimize the consequences of a violation on the security system.

Our work will focus in both software and data protection using both prevention and detection mechanisms.

These mechanisms are applied using, among others, mostly firewalls and intrusion detection systems (IDS). In this work we study the technologies that implement these prevention and detection mechanisms. Concretely, we focus on security tools available for Linux systems.

However, in many cases the use of the tools that implement these security mechanisms is not trivial, due to two main reasons:

1. It is necessary a basic knowledge of how the systems work, like a minimal understanding of the network protocols, operating systems low-level operations, etc.
2. For each tool it is necessary a knowledge about how to use it and how to configure it, in order to obtain the maximum performance in both efficacy and effectiveness of the tool.

Despite the obstacles posed by the first reason are not easy to overcome, as the basic knowledge required to know what is being done is indispensable, we find that it is possible to overcome the obstacles posed by the second of the reasons. Here we find the main motivation of our work. The difficulty using these tools is given by the fact that its configuration is usually carried out using a big pool of shell commands, configuration scripts and configuration files.

Therefore, to facilitate the work of the security administrator of Linux systems, in this work we present the design and implementation of a tool that allows the management of the probably most popular security tools for Linux: Snort and Iptables. We have

developed the application in order to allow security administrators to configure and manage such systems in a simple, convenient and efficient form. This tool allows the administrator to focus on the critical security tasks to be carried out, trying not to worry about the way they are performed.

As part of this thesis, we have analyzed and studied a set of different security mechanisms, which are briefly introduced as follows (reader can find more information in Section 2.1):

- Firewall systems allow performing access control of packets at the network layer. Thus, for each network packet that arrives to the computer, the firewall is able to decide, according to an established policy, the action to be performed on this packet. This action may be, basically, one the following: allow the network package continuing to its destination, discard the network packet or log the event in some operating system log file. As mentioned earlier, IPtables is the studied tool which applies this mechanism. This is a tool that is preinstalled on most Linux distributions and is widely used in both academic (teaching and research) and business environments.

- Intrusion Detection Systems (IDS) are software or hardware tools that analyze network packet traffic, system calls, application logs, etc., in order to look for evidence of the existence of an intrusion. If so, they alert the security administrator of the existence of some agent or element in the system which might involve an intrusion attempt, i.e., any attempt to compromise the security of the system. The analysis can be performed at the host or network level, differentiating both types of analysis by the field in which they are applied. Also there are different ways of carrying out the detection, being the IDS classified as signature based or anomaly based. Snort is the studied tool which it applies this mechanism, which is a free distribution IDS for Linux systems.

- An Intrusion Prevention Systems (IPS), in addition to alert the security administrator of the existence of some agent or element in the system which can be an intruder, allows taking some active action to avoid the malicious agent or restrict the possible actions you can take out the same. These systems are very similar to those presented earlier. Moreover, it can even be considered that they are the same type of system, but with the ability to respond to any intrusion attempt. Snort inline is the studied tool which it implements this mechanism.

- Integrity Control Systems allow to perform integrity checks into files to detect any alteration of its content, even though this modification should not have been conducted. These systems are usually applied on critical system files, like password storage files, operating system configuration files, etc. Advanced Intrusion Detection Environment (AIDE) is the studied tool which it applies this mechanism.

After the analysis and study of the abovementioned systems, we have taken the decision of implementing in our application the configuration and management of the Snort intrusion detection system, and the definition of rules for the firewall IPtables.

There are other applications on the market that allow you to manage these tools. For the management of IPtables, it can be found applications like phpIPtables and PHP Firewall Generator, while for managing Snort we can find applications like Analysis Console for Intrusion Database (ACID), Basic Analysis and Security Engine (BASE) and SNEZ.

All of them reveal some areas for improvement, which have been analyzed and included in our application with the goal of improving its quality.

Regarding the most important aspects included in the application requirements, the following are noteworthy:

- It will be a web application, due to the benefits provided by this type of applications, since it allows accessing to the application from any remote system with Internet access or it does not require that the computer on the application is running counts with a graphical environment.
- The application must mange two security tools (IPtables and Snort) whose purpose, as already mentioned, is completely different.
- The Snort management will be done through the possibility of configuring some of their preprocessors and some of its configuration variables.
- It will be possible to view the Snort detection rules which are being applied.
- You can view the alerts generated by Snort through the BASE tool. This tool allows you to view all alerts, allows the creation of graphs, provides alerts statistics, etc. The IPtables management will be done by setting all tables providing this tool. This includes the management of the various chains of each table and the rules of each of these chains.

In order to implement the application, there are a huge variety of programming technologies that provide different advantages. Therefore it is necessary to carry out a study of several of these technologies in order to identify their weaknesses and advantages and to help in taking the decision of what technology should be used. In the particular case of this work, we studied the use of Java Enterprise Edition (known as JEE), PHP Hypertext Pre-processor (known as PHP) and the Grails framework. Among the issues considered when choosing one of them, it has been considered:

- The experience that we have in the development of applications with each of these technologies. Since that we have never worked with some of these technologies, the learning time of each of them is a critical factor when to consider before choosing the technology to deploy the application, so as to avoid investing much time in learning both the syntax as how the technology works.
- Whether the technology allows developing modular programming. To perform a modular and structured programming is essential both for maintenance and for subsequently updates or improvements of the application.

- Whether the technology is multiplatform, namely, if the application developed with the technology could be executed independently of the environment in which this is accomplished.

- If there is a development environment for each technology. This is very useful because the development environments often automate repetitive tasks, such as creating classes, methods, constructors, etc.

We have study all these features and evaluated the proposed technologies. Consequently, we have considered that the most appropriate for carrying out this work is JEE.

After we have collected information about the application functionality requirements, and how to implement it, we are able to design of it. As we said before, the fact that this design is developed in a modular way is convenient for both the implementation complexity and to have an easy maintenance and updating of the application in the future. This is because each module in which we divide the application must be independent of the rest, so that any change in one of them does not affect the functionality provided by others. This modular design allows incorporate in future new functionality to configure other security tools, which is necessary in a world where technology changes continuously. To meet this objective, we use one of the most popular software architectures when developing web applications: the architecture Model-View-Controller (MVC). This is an architecture in four layers, each of which performs a well-defined work in the application operation:

- The Model layer is responsible for maintaining the data structure needed in order to allow the application perform its activities properly.

- The View layer is responsible for show the user the information collected through the data model as well as to visually present any possible action that can be carried out with the application.

- The Controller layer is responsible for receiving and processing requests made by the user, therefore selecting the proper action to be done based on the request and the data received. Then, after receiving the response of the action completed, it selects the next view that should be displayed to the user.

- The Services layer is responsible of storing all the possible actions that can be done with the system.

Within these four layers in which the application architecture is divided, we can build different modules which contain all the components necessary to meet certain with system functionality.

For example, within Services the layer, we set three modules, which are:

- A module that contains the components related to the functionality required for handling the tool Snort.

- A module that contains the components related to the functionality needed to manage the IPtables tool.

- A module to contains the components related to the general configuration of the application itself.

A critical point of the application is the design of the user interface. This is because this element is the one that the user perceives from the application and which he interacts with. Therefore, the two most important aspects which should aim to ensure are: the user must feel that the application is handled easily, and the user will not be forced to have to memorize how to perform each task (after all, this was one of the disadvantages detected in the task of configuring security tools through the operating system console or configuration files). To carry out a design that meets these characteristics, it is necessary clearly analyze three aspects:

1. The skills that the future users of the application will have. This makes us possible to design the user interface keeping in mind if these users have an advanced knowledge using certain software, if it is necessary to involve help options to show how to navigate the application, etc.

2. The tasks that users can perform through the application. This makes it possible to design the user interface keeping in mind the tasks that users will have to perform, therefore designing the structure of the elements appearing in the user interface in one way or another.

3. The characteristics that must satisfy the user interface of the application. Thus it is possible to design the user interface keeping in mind which are all characteristics that it must fulfill.

In addition to the identification of these three issues throughout the document, we also use ideas and concepts taken from the other external user interfaces for web applications. We mainly have considered thus applications that enable the management of software elements. In our case we have taken ideas from some web servers with a console management GUI and the BASE application. The advantage of this is that the end user may be familiar with these applications, so if our application looks like them, the user learning time would be definitely reduced.

During the implementation phase it is recommendable to carry out a set of tests to verify that the behavior of application components is the appropriate. In order to perform this test properly, thus corroborating the well performance of every component in the system, we have specified a testing plan to carry out during the implementation process of the application. In this plan we set out the following test levels:

- Unitary testing. This level of tests checks the operation of the internal components of the application, thus verifying whether the performance of each of these is correct.

- Integration testing. This level of tests makes possible to check if the assembly between several components of the application is done properly.

- System testing. This level of tests makes possible to check the operation of the entire application as a complete system.

For each of these levels we specify a set of different test cases, indicating for each case a number of attributes, mainly, what the test is, what input should be provided to the test, what output should be returned and what conditions must fulfill (both before as after the test is performed). If any of these tests do not meet all the given requirements, it would be necessary to perform the necessary changes in the implementation of the application until the behavior of the tested element becomes the correct.

Following the completion of the implementation phase of the application, and before finalizing the development, we carry out an evaluation of the final product. Before carrying out this evaluation it is necessary to specify the manner in which it will be performed. In this work, it is established that the assessment will consist of complete a series of simple tasks, which will be carried out by different users (with skills and capabilities matching those specified for future users of the application). Thus, it is possible to obtain information about whether they have met the proposed objectives in the beginning our work. Additionally, it is possible to get some ideas about potential application updates to be performed in the future, as it has finally happened in our case.

It should be mentioned that during the development process have appeared different problems. The most critical ones, and those that have provoked the biggest impact, are listed as follows:

1. During the realization of the planning work for the application development, the time that it would be possible to dedicate to this work during the course was overestimated. Concretely, we had considered five daily working hours, which has not been possible to carry out during the course, since I have gave priority to the degree ordinary subjects.

2. Some programming aspects that has been necessary solve had never been addressed, so it was necessary to spend some days longer than planned to implement this functionality in the application. This problem has increase to three months longer than planned development of the application.

As a final conclusion, after completing all the tasks necessary for the development of the application, we can say that this work meets all the goals set at the beginning of it, and as result it has produced a simple and intuitive application that will facilitate the management and configuration of security tools on Linux systems. Furthermore, since the application can be improved in some points, we propose future lines for updates. Some of them could be:

- Include the functionality necessary to create, erase and modify the detection rules of Snort.

- Include the functionality necessary to monitor activity generated by the IPtables tool, i.e., the logs.

- Include the functionality necessary to manage and monitor the activity of any other security tool, like the studied snort inline or AIDE.
- Include the functionality necessary to perform an access control on the tool.
- Include the functionality necessary to configure the application itself through the user interface.
- Performing a study of the simplicity and usability of the user interface of the application and, if needed, to make some improvements.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1.

Introducción y objetivos

1.1. Introducción

Hoy en día, uno de los elementos más importantes, por no decir el más importante, para cualquier usuario de cualquier sistema informático son los datos con los que éste sistema trabaja. Es de sobra conocido el gran crecimiento que ha sufrido Internet en los últimos años como consecuencia de los grandes avances que se han producido en las tecnologías de la información y en las comunicaciones. Dichas tecnologías han sido implantadas en multitud de organizaciones como pueden ser, por mencionar algunos ejemplos, en bancos, en comercios, en el mundo de la industria y de la administración pública, etc., con el objetivo fundamental de ayudar a los ciudadanos a realizar gestiones de todo tipo, desde hacer la compra hasta realizar trasferencias bancarias, necesitando para ello únicamente un ordenador personal y conexión a Internet.

Los sistemas informáticos de cualquier organización trabajan con una inmensa cantidad de datos confidenciales o, cuanto menos, privados tanto de usuarios como de la propia organización, a los cuales, debido a su naturaleza, no debe tener acceso cualquier usuario. Por tanto, parece evidente que existe la necesidad de proteger dichos datos[1]. Si atendemos a las principales conclusiones de la encuesta realizada por el Computer Security Institute (CSI) para los años 2010 y 2011, la cual se encuentra disponible en [23], pese a que las estadísticas muestran un avance en cuanto a seguridad de la información, destacan dos hechos: el aumento de complejidad de los ataques, en comparación con los ataques que se llevaban a cabo años atrás, y la ventaja que están tomando los atacantes debido a que las organizaciones tienden a exponer cada vez más sus procesos internos en Internet a través de aplicaciones web, y debido a que muchos de los desarrolladores de estas aplicaciones no llevan a cabo un proceso de desarrollo en el que se tenga en cuenta la seguridad, existen vulnerabilidades en estas aplicaciones que pueden ser aprovechadas por los atacantes.

Por otra parte, al mismo tiempo que se produce ese crecimiento y que los sistemas informáticos se vuelven más complejos, también crece el número de atacantes que buscan obtener un beneficio a través de sus ataques a sistemas informáticos, y la sofisticación de los mecanismos que permiten a estos atacantes perpetrar el ataque. Además de crecer el número de atacantes, es importante tener en cuenta que estos, actualmente, no necesitan

[1] Con el término protección de datos hacemos referencia al cumplimiento de los aspectos de confidencialidad, integridad y disponibilidad de los mismos.

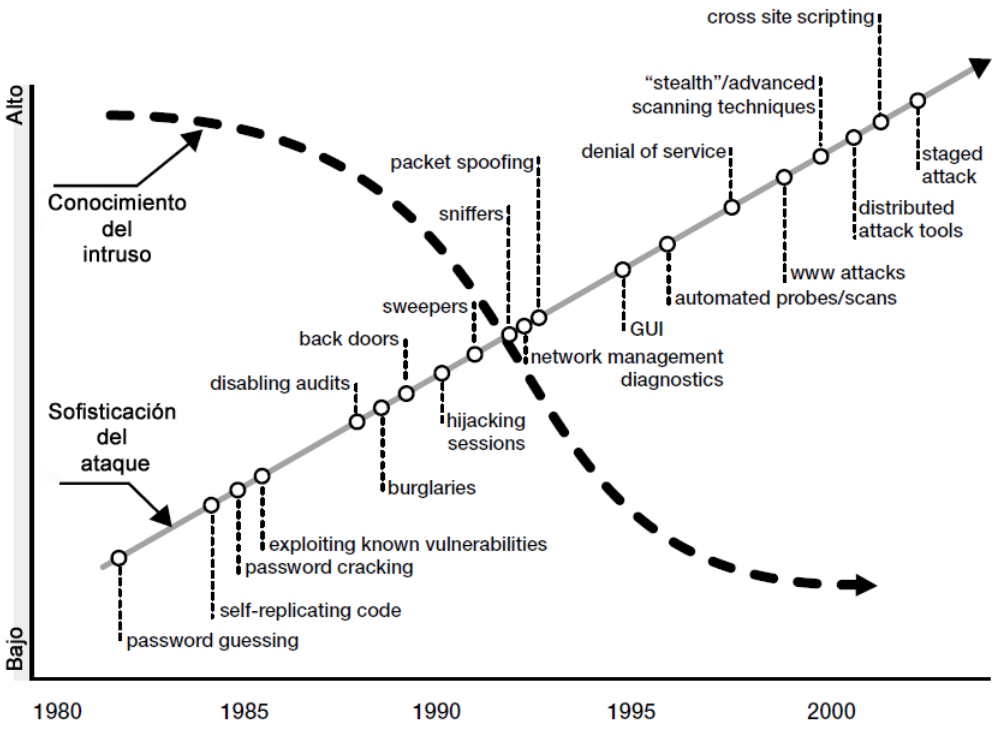

Figura 1.1.: Comparación del grado de sofisticación del ataque frente al grado de conocimiento requerido por parte del atacante a lo largo de los años.

contar con avanzados conocimientos de seguridad de la información ni de informática, ya que tienen a su disposición software que automatiza las tareas más comunes para violar la seguridad de sistemas informáticos. En la Figura (1.1), extraída de [2], se puede ver la relación entre la sofisticación de los ataques y el conocimiento de los atacantes a lo largo de los años. Para conocer algunas de las amenazas con las que pueden contar sistemas informáticos bancarios, militares, hospitalarios y de propósito general, se puede acudir a [3] donde se ofrece un análisis de algunas de las posibles amenazas de cada uno de los sistemas informáticos mencionados.

Para hacer frente a estas amenazas, un administrador de seguridad de un sistema de información tiene a su alcance distintos mecanismos de seguridad, los cuales pueden ser clasificados en función del momento en el que actúan:

- Mecanismos de prevención. Estos mecanismos tratan de minimizar la probabilidad de la ocurrencia de una violación del sistema de seguridad. Este tipo de mecanismo se aplica, por ejemplo, en los sistemas de cortafuegos que permiten llevar a cabo un control de acceso a nivel de la capa de red. Estos sistemas serán tratados más adelante en este documento.

- Mecanismos de detección. Estos mecanismos tratan de minimizar la propagación de una violación del sistema de seguridad. Este tipo de mecanismo se aplica, por ejemplo, en los sistemas para la detección de intrusiones, los cuáles también serán considerados más adelante en este documento.

- Mecanismos de corrección. Estos mecanismos tratan de minimizar el daño de una violación del sistema de seguridad. Este tipo de mecanismo se aplica, por ejemplo, en las herramientas que permiten establecer una honeypot[2].

- Mecanismos de recuperación. Estos mecanismos tratan de minimizar las consecuencias de una violación del sistema de seguridad. Este tipo de mecanismo se aplica, por ejemplo, a través de la creación de copias de seguridad de los datos con los que trabaja el sistema.

Parece evidente que la mejor manera de evitar que un atacante se lucre con el ataque a un sistema informático es, por una parte, previniendo que pueda llevar a cabo el ataque, y por otra parte, en caso de que la prevención falle, detectando que se está llevando a cabo una labor no deseada en el propio sistema para poder "expulsar" al elemento que está llevando a cabo dicha labor. También es importante notar que resulta más productivo prevenir o detectar una violación, que tener que desplegar mecanismos para la corrección y recuperación ante la violación ocurrida. Sin embargo, queremos aclarar que de ninguna forma se está queriendo decir con esto que tanto los mecanismos de corrección como los de recuperación no sean útiles a la hora de atajar ataques a un sistema informático, o haya que prestarles menor atención a este tipo de mecanismos.

En los sistemas Linux, los administradores de seguridad que emplean las herramientas que implementan los mecanismos de prevención y detección, como pueden ser Snort e IPtables, deben enfrentarse a realizar la configuración y monitorización de la actividad de las mismas a través de: la consola del sistema operativo, ficheros de configuración (los cuales suelen ser bastante extensos) o una combinación de los dos casos anteriores.

Esto puede hacer que la labor del administrador sea tediosa, necesitando memorizar gran cantidad de comandos de consola, de los posibles parámetros de configuración de cada aplicación, de los diferentes valores que puede especificar para cada uno de esos parámetros, etc., teniendo que acudir en varias ocasiones a los manuales de cada aplicación y buscar aquel elemento que desee configurar. Todo esto se puede ver agravado en el caso en el que el administrador nunca haya trabajado con dichas herramientas. Una de las principales consecuencias que pueden acarrear estos hechos es que el administrador cometa, de forma involuntaria, errores a la hora de configurar las herramientas, y tenga una falsa sensación de que el sistema se encuentra debidamente protegido, cuando en realidad no es así.

Debido a los motivos expuestos anteriormente, este trabajo tratará de desarrollar una herramienta que permita a los administradores de seguridad de sistemas Linux llevar a cabo su labor de forma más sencilla. Para esto, se tratará de lograr que el administrador se abstraiga de la mayor parte posible de los detalles de las herramientas que no contribuyan a alcanzar una mejor configuración del sistema. Con esto nos referimos a que el administrador pueda olvidarse completamente de las largas secuencias de comandos que tiene que introducir para configurar cada herramienta o de los diferentes valores que puede especificar para cada parámetro de configuración de cada una de ellas. De esta

[2] Se denomina honeypot tanto al software como al hardware que permite simular sistemas vulnerables, cuyo fin es atraer la atención y acciones de los atacantes.

forma, el administrador puede centrarse en su trabajo de defender el sistema sin tener que preocuparse de cómo llevará a cabo la configuración necesaria de cada herramienta de seguridad.

1.2. Objetivos

A la vista de lo expuesto en la sección anterior, el objetivo fundamental de este trabajo es el de desarrollar una aplicación de calidad para ayudar en la labor llevada a cabo por los administradores de seguridad del sistema operativo Linux. En base a ese objetivo principal, se proponen los siguientes objetivos parciales:

- Estudiar los principales mecanismos y herramientas de seguridad para decidir cuáles de ellos serán gestionados por la aplicación a desarrollar. Debido a la elevada cantidad de mecanismos y herramientas de seguridad disponibles, no resulta posible incluir funcionalidad para el manejo de todas ellas en la aplicación, por lo que es necesario acotar el alcance de la misma.
- Estudiar las posibles plataformas de desarrollo en las que puede ser implementada la aplicación. En este estudio se recogerán las principales ventajas e inconvenientes de cada plataforma de desarrollo y se escogerá aquella que más ventajas aporte.
- Llevar a cabo el desarrollo de una aplicación para la gestión remota de herramientas de seguridad en sistemas Linux. La aplicación resultante debe ser fácil de actualizar y mantener, por lo que un diseño modular resulta prácticamente imprescindible para lograr ese objetivo.
- Crear una máquina virtual en la que se incluirá el entorno en el que operará la aplicación, para comprobar el correcto funcionamiento de la misma.
- Evaluar la aplicación desarrollada. Para estudiar si la aplicación cubre todas las necesidades, será necesario:
  - Lograr configurar y poner en marcha, desde la aplicación, las herramientas de seguridad gestionadas por la misma.
  - Simular un ataque a la máquina virtual en la que se encuentra instalada la aplicación.
  - Comprobar que las alertas y entradas en los registros del sistema operativo son las mismas que si las herramientas hubieran sido configuradas de forma "tradicional".
- Redactar la documentación relativa a todo el proceso de desarrollo de la aplicación. Como ya hemos mencionado, la aplicación debe resultar fácil de actualizar y mantener, por lo que una buena documentación de la misma también resulta fundamental para alcanzar ese objetivo.
- Completado el proceso de desarrollo de la aplicación, y puesto que en la actualidad existen otras herramientas con finalidad similar, se pretende que los puntos de diferenciación de nuestra herramienta con el resto sean:
  - La posibilidad de gestionar y monitorizar la actividad de varias herramientas de seguridad.

- La visualización de información de ayuda acerca de los valores que puede tomar cada parámetro de configuración de cada herramienta de seguridad.
- El aviso al usuario en caso de que haya introducido algún valor incorrecto para alguno de los parámetros de configuración.
- La facilidad de uso gracias a una interfaz sencilla y cuidadosamente diseñada para garantizar una rápida adaptación al uso de la aplicación.

1.3. Fases del desarrollo

En esta sección se presentan cada una de las fases del proceso de desarrollo que se han llevado a cabo durante el transcurso del trabajo. El objetivo que se persigue con esto es que durante la lectura de este documento el lector tenga presente el orden lógico de todas las tareas que se han llevado a cabo a lo largo del proyecto y pueda conocer en qué punto del desarrollo se encuentra.

- Fase de estudio del problema. Debido a que el objetivo primordial de este trabajo es desarrollar una aplicación para la monitorización y defensa de sistemas Linux, es necesario llevar a cabo un estudio de algunos de los principales mecanismos y herramientas de seguridad que se encuentran disponibles para realizar esta labor. Este estudio ha sido realizado durante la fase de estudio del problema, y gracias a él ha sido posible definir el alcance del sistema a desarrollar.

  Además de este estudio, también resulta necesario conocer la situación actual en cuanto a sistemas que cumplan el mismo cometido que el sistema que se desea desarrollar. Por tanto, en esta fase también se ha llevado a cabo un estudio acerca de las aplicaciones que se encuentran en el mercado cuya finalidad es la misma, o cuanto menos similar, a la del sistema que se deseaba desarrollar, siendo identificados tanto sus puntos débiles como sus puntos fuertes, con el fin de que el sistema final mejorara las características que proveen el resto sistemas.

- Fase de análisis. Tras haber sido identificado el punto desde el cual partíamos, se tuvo que establecer qué es lo que el sistema debía, o no debía, hacer, la manera en la que lo debía hacer, las condiciones que debía cumplir a la hora de llevar a cabo su trabajo y la tecnología con la que se desarrollaría el mismo. Todas estas cuestiones han sido recogidas durante la fase de análisis, gracias a la documentación de requisitos de usuario y requisitos de software, y el estudio y valoración de distintas tecnologías disponibles para la implementación del sistema.

- Fase de diseño. Una vez que se hubo identificado el problema y las características que el sistema a desarrollar debía cumplir, era el momento de diseñar el mismo. Fue durante esta fase de diseño cuando se especificó la arquitectura del sistema, los distintos módulos con los que contaría, los distintos componentes que tendría, las clases que permitirían que el sistema llevara a cabo su funcionalidad (incluyendo sus atributos y métodos) y los elementos y estructuración de la interfaz gráfica del sistema.

  A parte de todos estos diseños, también resultó necesario diseñar un plan de pruebas para que durante la implementación del sistema se pudiera comprobar que cada uno de los componentes operaba, tanto de manera individual como manera de colectiva, correctamente.

- Fase de preparación del entorno de trabajo. Antes de comenzar la implementación del sistema fue necesario preparar el entorno en el que operaría el mismo. Durante esta fase se llevó a cabo la creación de una máquina virtual, la instalación en

dicha máquina de un sistema operativo y después la instalación del software que el sistema a desarrollar permitía gestionar.

- Fase de implementación y pruebas. Llegados a este punto, era hora de comenzar a implementar el sistema, es decir, era hora de volcar el diseño realizado en la fase anterior a un lenguaje de programación que permitiera que el diseño se "materializase".

  Para verificar que el proceso de implementación era correcto también se aplicaron durante esta fase cada una de las pruebas que se especificaron en la fase de diseño. Cuando los resultados de estas pruebas no eran los esperados, fue necesario realizar ajustes en la implementación para que la funcionalidad fuera correcta.

- Fase de evaluación. Al llegar a esta fase el sistema se encontraba totalmente implementado, y se diseñó un sistema de evaluación que, mediante la aplicación de las tareas comunes a las que se ve sometido el sistema, permitía verificar el correcto funcionamiento del mismo.

- Fase de redacción de la memoria. Una vez que el sistema estuvo completo, se recogieron todos los datos relevantes de su desarrollo en este documento, el cual permite trazar todo el trabajo realizado durante el desarrollo del sistema.

1.4. Medios empleados

En esta sección se presentan cada uno de los medios, tanto hardware como software, que fue necesario emplear para el desarrollo del trabajo.

1.4.1. Medios hardware

Los medios hardware empleados para el desarrollo de este trabajo fueron:

- Ordenador. Prácticamente todo el trabajo realizado durante el desarrollo del proyecto ha sido realizado con el ordenador:
- Disco duro externo. En este dispositivo se almacenaron las copias de seguridad del trabajo realizado a lo largo de todas las fases del desarrollo del sistema.

1.4.2. Medios software

Los medios software empleados para el desarrollo de este trabajo fueron:

- VMWare Player 3.1.4. Software de virtualización con el que se creó la máquina virtual en la que se llevo a cabo el trabajo.
- Ubuntu 10.10. Sistema operativo instalado en la máquina virtual.
- Rational Software Architect / Eclipse Indigo. Entornos de desarrollo utilizados para la implementación del sistema.
- Snort 2.9.2. Herramienta de detección de intrusiones que el sistema desarrollado permite gestionar.
- BASE. Interfaz web para la visualización y gestión de alertas generadas por Snort.
- IPtables 1.4.4. Herramienta de cortafuegos que el sistema desarrollado permite gestionar.
- MySQL Server y MySQL Client. Sistemas gestores de bases de datos sobre los que se creará la base de datos que contiene las alertas generadas por Snort.
- Apache Tomcat 6.0.35. Servidor web a través del cual es posible acceder al sistema desarrollado.
- Mozilla Firefox 14.0.1. Navegador web para acceder a la interfaz de usuario del sistema desarrollado.
- Procesador de texto LYX. Procesador de texto que fue utilizado para redactar este documento.
- JabRef 2.7. Sistema gestor de bases de datos bibliográficas, usado para crear la base de datos de la bibliografía empleada en este documento.

- OpenProj. Herramienta utilizada para la creación del diagrama de Gantt que muestra la planificación de las tareas a desarrollar.

1.5. Estructura de la memoria

En esta sección se ofrece una visión de la estructura que mantiene este documento, con el objetivo de mostrar una idea global de cómo se irá exponiendo el contenido del mismo al lector. Para facilitar la comprensión del trabajo realizado se ha tratado de estructurar el documento siguiendo las fases presentadas en la sección 1.3, y por tanto se encuentra dividido en los siguientes capítulos:

- El capítulo 1 presenta una breve introducción del documento, los objetivos que se desean lograr a lo largo del trabajo, las fases de desarrollo que componen el trabajo y la estructura seguida por este documento.

- El capítulo 2, por una parte, presenta un análisis de los principales mecanismos y herramientas de seguridad que se encuentran disponibles para la monitorización y defensa de sistemas Linux.

  Por otra parte, este capítulo presenta un estudio y una valoración de las herramientas que se encuentran en el mercado.

- El capítulo 3 presenta la planificación inicial del tiempo para llevar a cabo las tareas necesarias para completar el proyecto y un análisis de los costes que supone la realización del mismo.

- El capítulo 4 presenta los requisitos y restricciones del sistema que se debe desarrollar.

  Este capítulo también presenta un estudio de varias alternativas de la tecnología que se puede emplear para el desarrollo del sistema, así como una valoración de todas estas alternativas y una elección justificada de aquella que se empleará finalmente.

- El capítulo 5 presenta, en primer lugar, el diseño de la arquitectura que debe seguir el sistema a desarrollar.

  En segundo lugar, este capítulo presenta el diseño de las clases, así cómo de los atributos y métodos que deberán tener cada una, que será necesario implementar en el sistema.

  Por último, este capítulo presenta las cuestiones relacionadas con el diseño de la interfaz gráfica del sistema.

- El capítulo 6, por una parte, presenta las pruebas que deben ser aplicadas sobre el sistema, en tiempo de implementación, para asegurar que los componentes del mismo funcionarán correctamente.

  Por otra parte, este capítulo propone un sistema de evaluación del sistema y el análisis de los resultados arrojados por la aplicación de la misma.

- El capítulo 7 presenta las principales conclusiones extraídas a lo largo del desarrollo de este trabajo.

- El apéndice A presenta un manual de usuario de la aplicación desarrollada.

Capítulo 2.

Planteamiento del problema

2.1. Análisis de mecanismos y herramientas de seguridad

2.1.1. Cortafuegos

Para facilitar las explicaciones y ejemplos que llevaremos a cabo en esta sección, imagine una subred, a la que denominaremos Red administrada, la cual se encuentra conectada a Internet según se muestra en la Figura (2.1).

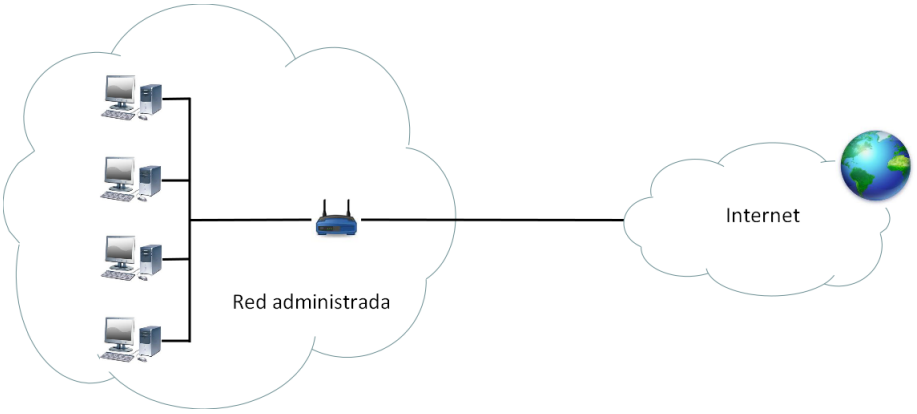

Figura 2.1.: Esquema de Red administrada.

Un cortafuegos o firewall es un sistema, o conjunto de sistemas hardware y software, que permite filtrar el tráfico de red. Es decir, es un sistema que permite inspeccionar los paquetes de red y, en función de si el paquete cumple o no ciertas características, hacer que prosiga su camino hacia la dirección de destino o no. Por tanto, con estos sistemas se puede conseguir la separación de una subred (en nuestro ejemplo denominada Red administrativa) del resto de subredes (en nuestro ejemplo Internet) las cuales no son confiables, consiguiendo así que sólo cierto tráfico del resto de subredes entre en la subred y viceversa.

De esta forma, sobre el esquema inicial de la Red administrada, un sistema de cortafuegos podría establecerse como en la Figura (2.2).

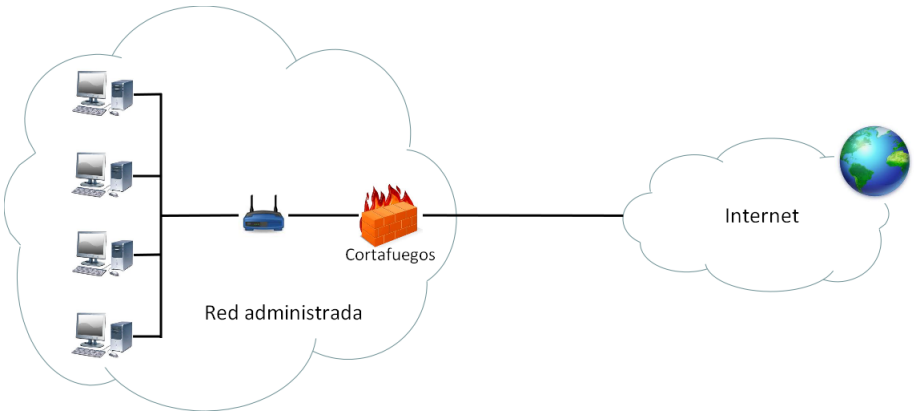

Figura 2.2.: Esquema de Red administrada con cortafuegos.

Podemos distinguir tres objetivos fundamentales de un cortafuegos:

- El primero es controlar todo el tráfico de entrada y salida de la red. Para que esto sea posible los cortafuegos suelen instalarse en el punto de acceso a la red.

  Existen otras maneras de utilizar cortafuegos como emplear varios niveles del mismo o cortafuegos distribuidos, pero la mejor opción en cuanto a su facilidad de mantenimiento es la mencionada en primer lugar.

- El segundo objetivo es permitir el paso únicamente al tráfico autorizado, de acuerdo con la política de seguridad establecida. Como todo el tráfico que entra y sale de la subred pasa a través del cortafuegos es posible que éste decida si dicho tráfico puede continuar hacia o desde la subred.

  Existen, básicamente, dos enfoques en cuanto a políticas de seguridad. La primera de ellas es "todo lo que no está explícitamente permitido, está prohibido", y la segunda es "todo lo que no está explícitamente prohibido, está permitido". Parece evidente que la primera es una política más segura que la segunda, sin embargo, en la decisión de tomar una u otra influyen, además de los motivos de seguridad, motivos administrativos de la organización propietaria de la subred y el valor de los activos que se están protegiendo.

- El tercer objetivo es conseguir un diseño de cortafuegos, y una instalación del mismo, de manera que éste no cuente con vulnerabilidades que permitan la penetración indebida. De no ser así, se estará confiando en que la subred se encuentra protegida, cuando realmente no es así. Esta cuestión está relacionada con el problema de que todos los sistemas de cortafuegos son centralizados, de esta forma si un atacante encuentra una vulnerabilidad en el mismo y consigue explotarla, el resto de la red, si no se encuentra debidamente protegida, correrá un grave peligro.

  A parte del correcto diseño e instalación, es necesaria una correcta configuración del cortafuegos para minimizar los riesgos de que éste falle; dicha configuración es responsabilidad del administrador. Aquí encontramos la motivación de nuestro

trabajo, puesto que el objetivo principal de la aplicación a desarrollar es facilitar el trabajo de dicho administrador, permitiendo que configure el cortafuegos de manera remota, ofreciéndole una interfaz gráfica para la configuración, etc.

Los cortafuegos pueden ser clasificados, según el tipo de filtrado de paquetes que llevan a cabo, en las siguientes categorías:

- Filtros de paquetes. En este tipo de filtrado, incluido en sistemas Linux y Windows, incluso en routers, cada paquete de red (también denominados datagramas) es examinado de manera individual y, para cada uno de ellos, se determina si debe pasar a través del cortafuegos o si por el contrario debe ser descartado. Las decisiones de si un datagrama atraviesa o no el cortafuegos suelen realizarse en base a la dirección IP de origen o destino, al puerto de origen o destino, al tipo de mensaje ICMP, etc.

  Este tipo de filtrado puede llegar a evitar ciertos tipos de ataques de denegación de servicio (aquellos que se basan en el envío de paquetes mal formados), de IP spoofing o bloquear IPs que son reconocidas como peligrosas.

  Las principales desventajas con las que cuenta este tipo de filtrado son la fragmentación de paquetes (que puede ser empleada para evitar el cortafuegos) o la dificultad de decidir qué direcciones IP de origen deben ser bloqueadas.

- Filtros de paquetes con memoria del estado. En este tipo de filtrado se controlan las conexiones TCP, ya que el cortafuegos es capaz de reconocer el inicio de una nueva conexión al observar el "three way handshake" (paquetes de SYN, SYNACK y ACK) realizado al comienzo de una nueva conexión TCP y el final de la misma al observar un paquete FIN (o al transcurrir cierto tiempo sin haber detectado actividad en la conexión).

  Este tipo de filtrado suele usarse para realizar filtrado de DNS o para dirigir cierto tipo de tráfico a aplicaciones de filtrado específicas.

  Las principales desventajas con la que cuenta este tipo de filtrado son: por una parte, no puede inspeccionar tráfico a nivel de aplicación (como, por ejemplo, código malicioso) y por otra parte, el mantener el estado de las conexiones establecidas supone un consumo de recursos, por lo que es vulnerable a ataques de denegación de servicio.

- Pasarelas de aplicación. Este tipo de filtrado se realiza mediante un servidor de aplicaciones, denominado pasarela de aplicación, el cual actúa como intermediario entre el usuario de la aplicación y la propia aplicación. El usuario, para utilizar cierta aplicación, sólo puede establecer una conexión con la pasarela, la cual puede llevar a cabo algún mecanismo de control de acceso sobre los usuarios que pueden usar dicha aplicación. Una vez autenticado el usuario, la pasarela se encarga de establecer la conexión con la aplicación y de redirigir el tráfico desde el usuario hacia la aplicación, y desde la aplicación hacia el usuario. Este tipo de filtrado suele ser complementado con filtros de paquetes.

La principal desventaja de este tipo de filtrado es que la pasarela de aplicación puede suponer un cuello de botella, debido a que es necesario establecer una pasarela para cada aplicación. Evidentemente, todo el tráfico dirigido hacia esta aplicación debe pasar por la pasarela de aplicación.

Explicado el concepto de cortafuegos tratemos cómo se realiza la configuración del filtrado que debe llevar a cabo el mismo. Esta configuración suele realizarse mediante un conjunto de reglas especificadas en una tabla. En dicha tabla se establecen dos tipos de campos: campos de condiciones que el paquete inspeccionado debe cumplir y campos de acciones a llevar a cabo en caso de existir una coincidencia. Entre los campos de condiciones podemos encontrar dirección IP de origen, dirección IP de destino, protocolo de transporte empleado, puerto de origen y puerto de destino. Las acciones que comúnmente se toman sobre los paquetes que cumplen las condiciones suelen ser: aceptar dicho paquete, descartar dicho paquete o registrar la llegada de dicho paquete. Cada tupla de la tabla es considerada una regla. Hay que tener en cuenta que el orden de las reglas dentro de la tabla es muy importante, ya que de existir la posibilidad de que un paquete cumpla las condiciones de varias reglas, se aplicará la acción asociada a la regla que se encuentre en primer lugar.

Vistos los conceptos más importantes de los cortafuegos, estudiaremos una aplicación que permite crear reglas de filtrado de paquetes y que se encuentra incluida en todas las distribuciones modernas de GNU/Linux: IPtables. La elección de esta herramienta en el ámbito de este trabajo se debe, principalmente, a los siguientes motivos: la gran difusión con la que cuenta la misma, con lo que ello supone (grandes cantidades de documentación: manuales, tutoriales, foros,... acerca de la herramienta), la facilidad a la hora de trabajar con ella (además al encontrarse incluida por defecto en la mayoría de las distribuciones de GNU/Linux, no es necesario su compilación e instalación) y la experiencia obtenida del uso previo de la misma.

IPtables

IPtables es una herramienta, incluida en sistemas Linux a partir de la versión 2.3.15 del kernel, que permite implementar políticas de filtrado de paquetes de red. El origen de esta herramienta se encuentra en herramientas más antiguas como ipfw, ipfwadm o ipchains. La funcionalidad de todas ellas es la misma que la de IPtables, de hecho esta herramienta es la última evolución de aquellas.

Es importante mencionar que esta herramienta se maneja mediante línea de comandos, es decir, no existe un fichero de configuración de IPtables en el que añadir, por ejemplo, políticas de seguridad o reglas de filtrado. Es decir, toda configuración de la herramienta debe realizarse mediante un terminal de comandos o mediante scripts. Esto es necesario tomarlo en cuenta, puesto que al reiniciar el sistema, si no se ha creado un script que se ejecute al arranque para configurar IPtables, se perderán todas las reglas y políticas creadas de forma previa al reinicio del sistema.

IPtables se basa en el uso de reglas agrupadas en cadenas, la cuales, a su vez, se encuentran agrupadas en tablas. A continuación se muestran, de manera esquemática,

las tablas y cadenas con las que cuenta la herramienta, dando una breve descripción del uso de cada una de ellas:

- Filter. Su finalidad es almacenar las reglas de filtrado, que como explicamos antes, son las encargadas de "decidir" qué paquetes de red tienen permiso para entrar o salir de la subred protegida mediante cortafuegos. Esta tabla dispone de las siguientes cadenas:
  - Input. Su finalidad es almacenar las reglas de filtrado que permiten "decidir" qué paquetes de red están autorizados a entrar en la subred.
  - Output. Su finalidad es almacenar las reglas de filtrado que permiten "decidir" qué paquetes de red están autorizados a salir de la subred.
  - Forward. Su finalidad es almacenar reglas de filtrado que permiten "decidir" qué paquetes de red están autorizados a ser encaminados hacia otra máquina.
- Nat (Network Address Translation). Su finalidad es almacenar las reglas que permiten realizar un enmascaramiento de la subred que se encuentra por detrás del cortafuegos. Esta tabla dispone de las siguientes cadenas:
  - Output. Su finalidad es almacenar reglas que permiten tratar la dirección de red antes de que el paquete de red sea enviado a través de una interfaz de red.
  - Prerouting. Su finalidad es almacenar reglas que permiten tratar la dirección de red antes de haber tomado las decisiones de reenvío del paquete.
  - Postrouting. Su finalidad es almacenar reglas que permiten tratar la dirección de red después de haber tomado las decisiones de enrutado.
- Mangle. Su finalidad es almacenar las reglas que permiten realizar modificaciones sobre los parámetros de los paquetes de red. Algunos de los parámetros que podemos modificar son TOS (Type Of Service) o TTL (Time To Live). Esta tabla dispone de las siguientes cadenas:
  - Input. Su finalidad es almacenar reglas que permiten modificar paquetes de red que tratan de entrar en la subred.
  - Output. Su finalidad es almacenar reglas que permiten modificar paquetes de red que tratan de salir de la subred.
  - Forward. Su finalidad es almacenar reglas que permiten modificar paquetes de red que tratan de ser encaminados hacia otra máquina.
  - Prerouting. Su finalidad es almacenar reglas que permiten modificar paquetes de red antes de haber tomado las decisiones de reenvío del paquete.
  - Postrouting. Su finalidad es almacenar reglas que permiten modificar paquetes de red después de haber tomado las decisiones de reenvío.

- Raw. Su finalidad es configurar excepciones en el seguimiento de paquetes, es decir, permite evitar que IPtables inspeccione ciertos paquetes de red. Esta tabla dispone de las siguientes cadenas:
  - Output. Su finalidad es almacenar reglas que permiten evitar la inspección de paquetes de red que tratan de salir de la subred.
  - Prerouting. Su finalidad es almacenar reglas que permiten evitar la inspección de paquetes de red antes de haber tomado las decisiones de reenvío del paquete.

Además de estas cadenas especificadas, la herramienta ofrece la posibilidad de que el usuario cree nuevas cadenas.

La manera en la que IPtables busca alguna coincidencia entre el paquete que debe inspeccionar en ese momento y las reglas especificadas es la siguiente: si el paquete proviene del exterior de la subred, es decir, se trata de un paquete de entrada, utiliza las reglas almacenadas en la cadena Input para decidir si el paquete está autorizado a continuar o no. Si este tiene autorización y además debe ser reenviado hacia otra máquina, emplea las reglas almacenadas en la cadena Forward para decidir si está autorizado a continuar. Por el contrario, si el paquete proviene de la subred, es decir, se trata de un paquete de salida, utiliza las reglas almacenadas en la cadena Output para decidir si el paquete tiene autorización para continuar hacia el exterior de la subred o no. En la Figura (2.3) se muestra el proceso explicado mediante un diagrama.

Una vez que IPtables ha encontrado una coincidencia entre una regla y el paquete inspeccionado, aplica la acción que el administrador haya especificado para ese tipo de paquetes de red. Las acciones que se pueden llevar a cabo son las siguientes:

- Accept. Esta acción permite el paso de paquetes de red.
- Reject. Esta acción permite bloquear el paso de paquetes de red, enviando una notificación al origen mediante el protocolo ICMP.
- Drop. Esta acción permite bloquear el paso de paquetes de red. En este caso no se envía ninguna notificación al origen.
- Masquerade. Esta acción permite el enmascaramiento de la dirección de red de origen de manera dinámica.
- Dnat. Esta acción permite el enmascaramiento de la dirección de red de destino.
- Snat. Esta acción permite el enmascaramiento de la dirección de red de origen de manera estática.
- Log. Esta acción permite almacenar, en un registro, la coincidencia de un paquete de red con esta regla. Si se emplea esta acción, hay que tener en cuenta que, a diferencia de los demás casos, IPtables seguirá buscando coincidencias entre las demás reglas y el paquete que ha hecho saltar esta regla.

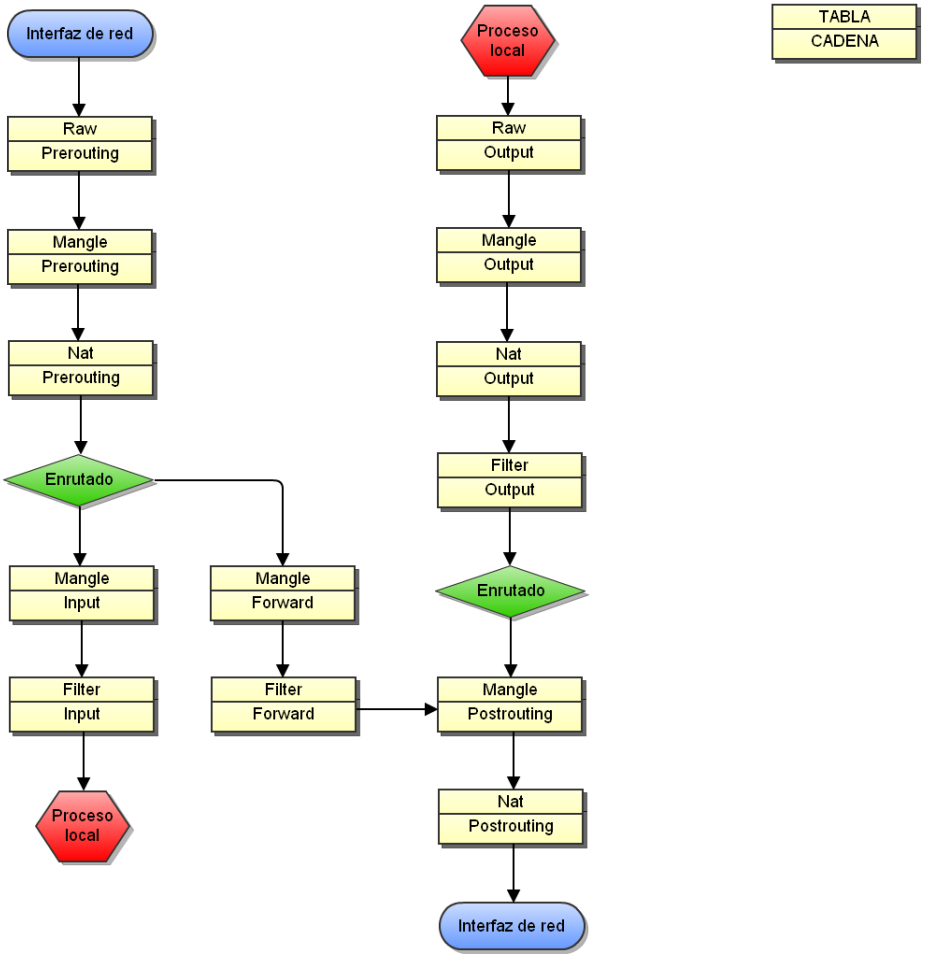

Figura 2.3.: Diagrama de funcionamiento de la búsqueda de coincidencias llevada a cabo por IPtables.

A cada una de las cadenas se le aplica una acción por defecto. Esto es así ya que en caso de encontrar un paquete de red al que no se le puede aplicar ninguna regla, se le aplica la regla que tenga la cadena.

Para obtener una descripción más detallada sobre cortafuegos e IPtables se puede consultar [3], [11], [15], [17], [12] y [4].

2.1.2. Sistemas de detección de intrusiones (IDS)

A lo largo de esta sección, haremos referencia en varias ocasiones al término intrusión, por lo que, en primer lugar, definiremos el mismo. Según [15] este término hace referencia a "un conjunto de acciones que intentan comprometer la integridad, confidencialidad o disponibilidad de un recurso". Es decir, una intrusión abarca desde un acceso no autorizado a una máquina hasta una denegación del servicio que proporciona la misma. Por tanto, denominaremos como sistemas de detección de intrusiones (IDS, Intrusion Detection System) a aquellos sistemas cuyo objetivo es detectar signos de actividad intrusiva.

Existen varios enfoques para clasificar IDSs. Atendiendo a cuál es el ámbito del IDS, se pueden establecer las siguientes categorías:

- IDS de host (HIDS). Detecta actividad sospechosa en un único sistema (ya sea este un servidor, un computador, ...). Por ejemplo, una forma común es analizar las llamadas al sistema que realizan las aplicaciones monitorizadas en busca de flujos de información sospechosos.

- IDS de red (NIDS). Busca intrusiones en los paquetes de red. En este caso el IDS puede situarse en cualquier máquina de la subred (sea ésta un computador, un router o un switch), no siendo necesario que se sitúe en todas y cada una de las máquinas que conformen la subred.

Atendiendo a cómo llevan a cabo la funcionalidad explicada anteriormente, se pueden establecer las siguientes categorías:

- Detección basada en firmas (Misuse detection). Estos sistemas operan siguiendo un modelo del comportamiento de un intruso, aprovechando que muchos ataques siguen un patrón determinado, denominado firma. Una firma no es más que una serie de reglas que definen un ataque de intrusión. Por tanto, manteniendo una base de datos con firmas de ataque se puede comprobar si los paquetes de red que llegan al IDS coinciden con alguna de esas firmas y son, por tanto, potencialmente peligrosos. Detectado este hecho, se actuará conforme el IDS haya sido configurado: lanzando una alerta, avisando al administrador, registrando lo ocurrido en un registro del sistema, etc.

  Aunque el administrador del IDS puede añadir nuevas firmas a la base de datos o modificar las ya existentes en ella, la principal desventaja con la que cuentan este tipo de sistemas es que se requiere conocer el ataque para poder generar una firma capaz de detectarlo, por lo que estos sistemas son poco útiles frente a nuevos ataques o ataques de día cero (también conocidos como ataques 0day).

- Detección basada en anomalías (Anomaly detection). Estos sistemas operan, en contraposición con los anteriores, sin contar con un patrón de comportamiento del intruso. Este método de detección es mucho más complicado puesto que supone crear un perfil de tráfico de buen comportamiento, observando durante cierto periodo de tiempo, la actividad normal del sistema o el tráfico de red normal. Para

realizar esta labor, se suelen emplear técnicas de inteligencia artificial, como por ejemplo, redes de neuronas artificiales.

La principal desventaja con la que cuentan este tipo de sistemas es la gran complejidad para elaborar un modelo estadístico de comportamiento usual.

Por otro lado, la ventaja que presentan con respecto a los sistemas anteriores es que en este caso no es necesario conocer el ataque antes de que éste se produzca, ya que permite detectar ataques no incluidos en la base de datos de firmas.

En la sección anterior hemos visto cómo un cortafuegos tiene la capacidad de inspeccionar los campos de cabecera de los paquetes de red y decidir qué paquetes pueden atravesar el mismo. Sin embargo, para detectar muchos ataques es necesario realizar una inspección de los datos que transportan los paquetes de red. Es aquí donde entran en juego los NIDS puesto que estos dispositivos permiten realizar una comparación del contenido (payload) de los paquetes de red con una serie de patrones, y decidir si dicho paquete supone un peligro para el sistema o no. Llegados a este punto, es conveniente realizar una distinción entre sistemas de detección de intrusiones y sistemas de prevención de intrusiones (IPS, Intrusion Prevent System). Este último tipo de sistemas permiten detectar tráfico potencialmente malicioso de la misma forma que los IDS, pero además, permiten filtrar dicho tráfico[1]. Entre los ataques que un NIDS permite detectar encontramos los siguientes: mapeado de red, escaneo de puertos, escaneo de la pila TCP, ataques de denegación de servicio (DoS), ataques de vulnerabilidades del sistema operativo, ataques de vulnerabilidades de aplicación, gusanos y virus.

Independientemente de la categoría en la que se encuentre clasificado un IDS, éste debe cumplir una serie de propiedades:

- Debe ejecutarse continuamente, sin necesidad de ser supervisado. No es factible disponer de una persona encargada de vigilar todo el tráfico de la red, comprobando que, efectivamente, el IDS cumple con su obligación.
- Debe poder adaptarse a los cambios en el sistema. Existen muy pocos sistemas que se mantienen inalterados con el paso del tiempo, por lo que el IDS debe ser capaz de adaptarse a los cambios que se produzcan en el sistema.
- Debe presentar tolerancia a fallos y capacidad de respuesta ante situaciones inesperadas.
- No debe suponer gran sobrecarga del sistema. El uso del IDS no debe suponer una gran pérdida de rendimiento para el resto de aplicaciones del sistema, puesto que, de ser así, el trabajo que realice puede llegar a ser poco útil.
- No debe generar numerosos falsos positivos. Si el IDS, en algún momento de su utilización, comienza a generar gran cantidad de falsos positivos (es decir, acciones detectadas como intrusiones que en realidad no son tal) se puede llegar a desconfiar de su trabajo y dejar de lado su uso.

[1] Más adelante se realizará un análisis de los sistemas de prevención de intrusiones.

Para entender qué son los falsos positivos nos ayudaremos de la Tabla (2.1), la cual ofrece una clasificación en función de si una actividad es detectada, o no, como una intrusión, cuando dicha actividad en realidad supone, o no, una intrusión. De esta clasificación se obtienen cuatro posibles casos:

- Verdadero positivo (VP). Se consideran así las actividades que son detectadas como intrusiones y que en realidad sí lo son.
- Falso negativo (FN). Se consideran así las actividades que no son detectadas como intrusiones, pero que en realidad sí lo son.
- Falso positivo (FP). Se consideran así las actividades que son detectadas como intrusiones, pero que en realidad no lo son.
- Verdadero negativo (VN). Se consideran así las actividades que no son detectadas como intrusiones y que en realidad no lo son.

A partir de esta clasificación se puede establecer una tasa de aciertos (más conocida por su término en inglés como hit rate) y una tasa de falsas alarmas (más conocida por su término en inglés como false alarm rate). Cada una de estas tasas se calcula de la siguiente manera:

$$Hit\,rate = \frac{VP}{VP + FN}$$

$$False\,alarm\,rate = \frac{FP}{FP + VN}$$

Parece evidente que el objetivo es maximizar la tasa de aciertos y al mismo tiempo minimizar la tasa de falsas alarmas.

Tabla 2.1.: Clasificación de intrusiones frente a detección de las mismas.

INTRUSIÓN	**DETECCIÓN**	
	Sí	**No**
Sí	VP	FN
No	FP	VN

La principal limitación los IDS se encuentra en que, a pesar de que algunos ataques de intrusión son muy obvios, el caso general es que detectar intrusiones es complicado, por lo que, por el momento, no es lógico tratar de encontrar una solución "completa" al

problema. Para explicar mejor esto, hay que tener en cuenta que los fallos de seguridad pueden ser de dos tipos, aquellos fallos que producen un error y los fallos que no producen ningún error. Una buena práctica es tratar de diseñar los sistemas de manera que los fallos de seguridad fueran del primero de los tipos expuestos, sin embargo, esto no siempre es posible. Desgraciadamente, los ataques de intrusión caerían (la mayoría de las veces) en el segundo de estos tipos, por lo que su detección, al no provocar un fallo, es más complicada.

Vistos los conceptos más importantes de los IDSs, estudiaremos una de las implementaciones más populares de este tipo de sistemas: Snort. De forma parecida al caso de IPtables, la elección de esta herramienta en el ámbito de este trabajo se debe, principalmente, a los siguientes motivos: la gran difusión con la que cuenta la misma, con lo que ello supone (grandes cantidades de documentación: manuales, tutoriales, foros,... acerca de la herramienta) y la experiencia obtenida del uso previo de la misma. El uso de este software es algo más complejo que en el caso de IPtables debido a la gran cantidad de opciones con las que cuenta el mismo.

Snort

Snort es el IDS de red más implantado, que combina detección basada en firmas y detección basada en anomalías, y que se encuentra disponible para plataformas Linux, UNIX y Windows[2].

Esta herramienta puede ser configurada para ejecutar en tres modos de operación: modo sniffer, modo de registro de paquetes, modo sistema de detección de intrusiones de red. El modo sniffer permite mostrar por pantalla las cabeceras IP, TCP, UDP o ICMP, el contenido (payload) y las cabeceras del nivel de enlace de los paquetes de red capturados por Snort. El modo de registro de paquetes permite almacenar en disco (utilizando distintos formatos), creando una jerarquía de directorios basada en una de las direcciones IP contenidas en el datagrama, los paquetes de red recibidos. Además, como cualquier otro sniffer, permite especificar filtros de captura para obtener sólo aquellos paquetes de red que cumplan ciertas características. El modo sistema de detección de intrusiones de red, el cual es la razón de ser de Snort, permite inspeccionar los paquetes de red capturados usando las reglas almacenadas en el sistema para decidir si es necesario llevar a cabo alguna acción sobre dicho paquete.

Snort se basa en el uso de reglas, las cuales pueden escribirse en una única línea o en varias líneas anteponiendo el símbolo \ al salto de línea. Todas las reglas que emplea esta herramienta se encuentran divididas en dos secciones lógicas: cabecera y opciones. La cabecera de la regla establece algunas de las condiciones que debe cumplir un paquete para que sea considerado como sospechoso, así como la acción a llevar a cabo si se detecta el mismo. Estas condiciones se definen a través de:

- Acción. Indica a Snort lo que tiene que hacer en caso de existir una coincidencia entre el paquete de red inspeccionado con la regla que se está evaluando. Existen

[2] Para el correcto uso de la herramienta es necesario disponer de las librerías libcap y la librería Libnet (recomendable).

los siguientes cinco tipos de alertas: alert, log, pass, activate y dynamic. Además de estos tipos, la herramienta ofrece al usuario la posibilidad de crear otros nuevos.

- Protocolo. Indica el protocolo que tiene que seguir el paquete de red inspeccionado para que se produzca una coincidencia con la regla. Actualmente ofrece soporte para los siguientes protocolos: TCP, UDP, ICMP e IP.
- Dirección IP de origen. Indica la dirección IP de origen que debe contener el paquete de red inspeccionado para que se produzca una coincidencia con la regla. No es necesario especificar una única dirección, o conjunto de direcciones, ya que existen palabras reservadas para especificar cualquier dirección IP de origen.
- Puerto de origen. Indica el puerto de origen que debe contener el paquete de red inspeccionado para que se produzca una coincidencia con la regla. De la misma forma que en el caso anterior, no es necesario especificar un único puerto, o conjuntos de puertos, ya que también existen palabras reservadas para especificar cualquier puerto de origen.
- Operador de dirección. Indica la orientación del tráfico en la que se aplica la regla. Este operador se denota por el conjunto de símbolos "->" o "<>". El primero de ellos indica que aquello que se encuentra a la izquierda del símbolo se considera tráfico desde el origen y aquello que se encuentra a la derecha se considera tráfico hacia el destino. En el caso del segundo operador, ambos lados pueden considerarse como origen y destino[3].
- Dirección IP de destino. Indica la dirección IP de destino que debe contener el paquete de red inspeccionado para que se produzca una coincidencia con la regla. Igual que en el caso de la dirección IP de origen, no es necesario especificar una única dirección, o conjunto de direcciones, ya que existen palabras reservadas para especificar cualquier dirección IP de destino.
- Puerto de destino. Indica el puerto de destino que debe contener el paquete de red inspeccionado para que se produzca una coincidencia con la regla. De la misma forma que en ocurría con el puerto de origen, no es necesario especificar un único puerto, o conjuntos de puertos, ya que también existen palabras reservadas para especificar cualquier puerto de destino.

Las opciones de la regla establecen las partes del paquete que deben ser inspeccionadas, así como el mensaje que debe ser mostrado al administrador en caso de detectarse tráfico potencialmente peligroso. Existen, principalmente, cuatro categorías de opciones: generales, las cuales ofrecen información acerca de la regla pero no intervienen en el proceso de detección (msg, sid, rev, ...); relativas al contenido (payload), las cuales inspeccionan los datos contenidos en el paquete de red (content, rawbytes, pcre, ...); no relativas al contenido (non-payload), las cuales, como su propio nombre indica, no están

[3] En versiones más antiguas de Snort existía la posibilidad de emplear un tercer operador ("<-"), el cual fue eliminado para conseguir mayor consistencia a la hora de leer las reglas.

relacionadas con el contenido del paquete (fragoffset, ttl, tos, ...); y, por último, post-detección, las cuales especifican disparadores que saltan después de haberse producido una coincidencia (logto, session, count, ...).

La manera en la que Snort (trabajando en modo sistema de detección de intrusiones de red) busca alguna coincidencia entre el paquete que debe inspeccionar en ese momento y las reglas especificadas es la siguiente: en primer lugar, el decodificador de paquetes de la herramienta interpreta el contenido del paquete de red. Después se cede el control al preprocesador, el cual permite manipular el paquete o realizar cierto tipo de análisis sobre el mismo. Este módulo de la herramienta fue añadido a partir de la versión 1.5 con la finalidad de hacer más sencilla la labor de añadir plugins para aumentar la funcionalidad de Snort (hay que tener en cuenta que es posible añadir varios preprocesadores). Una vez que los preprocesadores han acabado su labor, se aplica el filtro de reglas de la herramienta que decidirá si un paquete es potencialmente peligroso o no lo es. Llegados a este punto es necesario explicar que es altamente recomendable que este módulo compruebe únicamente reglas que son estrictamente necesarias. Explicamos esto porque mientras el IDS se encuentra decidiendo si el paquete de red coincide con algún patrón de los que tiene almacenados, dejará pasar el resto de los paquetes de red, es decir, Snort no detendrá el tráfico de red que no sea capaz de procesar sino que simplemente permitirá su paso (independientemente de si este tráfico supone una amenaza o no). Por último, se llevan a cabo las acciones asociadas a cada caso: si el paquete no contiene ningún patrón sospechoso se permite su paso, y si contiene un patrón sospechoso se lanzan las acciones oportunas (aviso al administrador de la red, añadir incidencia en la base de datos, ...).

Para obtener una descripción más detallada sobre IDSs y Snort, se puede consultar [3], [15], [17] y [25].

2.1.3. Sistemas de prevención de intrusiones (IPS)

En [24] se puede encontrar un análisis muy completo acerca de, entre otras muchas cosas, los principios de un sistema de prevención de intrusiones (IPS, Intrusion Prevention System), qué tecnologías existen para implementarlos o cuál es la forma en la que estos sistemas trabajan.

Como ya hemos visto, los sistemas de detección de intrusiones llevan a cabo un proceso de monitorización y análisis de eventos, ocurridos bien en un computador concreto o bien en una red de computadores completa, en busca de signos de violaciones o posibles violaciones de la seguridad del sistema. Un sistema de prevención de intrusiones ofrece las mismas propiedades que un sistema de detección de intrusiones y añade características para tratar de detener el ataque que se esté llevando a cabo. Al ofrecer las mismas características que un sistema de detección, los sistemas de prevención de intrusiones suelen ser identificados con las siglas IDPS (Intrusion Detection and Prevention System).

Puesto que las principales características de los sistemas de detección de intrusiones ya fueron expuestas en la sección 2.1.2, y debido a que los sistemas de prevención de intrusiones cuentan, básicamente, con las mismas características que aquellos, evitamos repetir toda la información que se encuentra disponible en la mencionada sección.

Vistos los conceptos más importantes de los IPSs, estudiaremos una de las implementaciones más populares de este tipo de sistemas: Snort_inline. Como en ocasiones anteriores, la elección de esta herramienta en el ámbito de este trabajo se debe, principalmente, a la gran difusión con la que cuenta la misma, con lo que ello supone (grandes cantidades de documentación: manuales, tutoriales, foros,... acerca de la herramienta).

Snort_inline

Snort_inline es una versión modificada de la herramienta Snort que vimos anteriormente. Se emplea como un complemento de este último, y la forma de trabajar de la herramienta es la siguiente: los paquetes de red que llegan al sistema son inspeccionados, en primer lugar, por el cortafuegos (IPtables), el cual tras decidir si el paquete tiene permiso para continuar lo ubica en una cola. Snort_inline toma los paquetes de dicha cola, los inspecciona en busca de patrones que lo identifiquen como tráfico potencialmente peligroso, siguiendo el mismo esquema de trabajo que identificamos al documentar Snort, y el paquete inspeccionado es etiquetado en función de si es considerado como tráfico sospechoso o no. Después, vuelve a tomar el control del paquete de red el cortafuegos, el cual, en función del etiquetado que contenga el paquete, descartará el mismo o lo dejará continuar hacia el destino.

En cuanto a su modo de empleo, no es muy diferente del modo de empleo de la herramienta Snort. Simplemente se debe tener en cuenta que en el cortafuegos utilizado, al utilizarse como un primer filtro, los paquetes que tengan permiso para continuar deberán ser redirigidos a la cola de entrada de la herramienta Snort_inline, el cual utilizará reglas que siguen el mismo formato que las reglas de Snort, para decidir si un paquete de red tiene permiso para continuar su recorrido. La única diferencia en las reglas de esta herramienta con respecto a Snort las encontramos en el campo de acción de la cabecera de las mismas, ya que existen, además de los ya mencionados con anterioridad, los siguientes tipos: drop, reject o sdrop.

Se puede decir que la versión de Snort presentada anteriormente corresponde al concepto de sistema de detección de intrusiones, ya que únicamente ofrece la posibilidad de, habiendo detectado un paquete cuyo contenido es susceptible de entrañar algún peligro, avisar al administrador del sistema de la presencia de dicho paquete. Sin embargo, Snort_inline va un poco más allá, correspondiendo al concepto de sistema de prevención de intrusiones, ya que además de ofrecer la posibilidad de avisar al administrador al detectar tráfico sospechoso, permite realizar acciones en función del tráfico inspeccionado (bloqueando tráfico sospechoso, por ejemplo).

Para obtener más información acerca del funcionamiento de Snort_inline se puede consultar [20]. En [21] se puede encontrar un artículo en el que se expone el uso de esta herramienta en distintos ámbitos.

2.1.4. Control de integridad

El objetivo perseguido por el mecanismo de control de integridad es detectar qué ficheros o directorios del sistema han sido modificados, aun cuando éstos deberían permanecer inalterados. Este objetivo resulta ser fundamental para mantener la seguridad de un sistema, puesto que algunas personas pueden llevar acabo ataques cuyo fin sea modificar ciertos ficheros para ocultar la actividad ilícita que están realizando.

El ejemplo más claro que muestra la importancia del control de integridad lo encontramos en el caso en el que un atacante haya logrado permisos de administrador e instalado en el computador un software malicioso para lograr acceso privilegiado de manera continua, al mismo tiempo que oculta su presencia en el sistema (este tipo de software suele denominarse rootkit). Dándose este caso, lo más probable es que el atacante haya sustituido algunos ficheros binarios, bien pertenecientes al sistema operativo o bien pertenecientes a aplicaciones de nivel de usuario, para evitar que su actividad maliciosa sea detectada. Algunos de estos ficheros podrían ser los que proveen la funcionalidad de comandos como ifconfig, df o who, entre muchos otros. De esta forma, el administrador del sistema víctima se vería obligado a convivir con el intruso, sin ser el primero consciente de que alguien malintencionado está haciendo uso de los recursos del sistema que administra para diversos fines, como podría ser realizar descargas de archivos o realizar ataques a otras máquinas. Estos últimos son los conocidos como ataques de denegación de servicio (DoS, por sus siglas en inglés de Denegation of Service).

Vistos los conceptos más importantes, así como la importancia del control de integridad, estudiaremos una de las implementaciones más populares de este tipo de sistemas: AIDE. Como en los demás casos, la elección de esta herramienta en el ámbito de este trabajo se debe, principalmente, a la gran difusión con la que cuenta la misma, con lo que ello supone (grandes cantidades de documentación: manuales, tutoriales, foros,... acerca de la herramienta).

Advance Intrusion Detection Environment (AIDE)

Como podemos leer en [18], Advance Intrusion Detection Environment (AIDE) es una herramienta de control de la integridad. Dicho de otro modo, esta herramienta se encarga de realizar comprobaciones de la integridad de ficheros y/o directorios, es decir, que los ficheros y/o directorios controlados no sufran ningún tipo de alteración (ya sea ésta modificación o borrado de parte o totalidad del contenido o propiedades de los mismos).

La forma que tiene AIDE (y en general la mayoría de herramientas destinadas a realizar control de integridad) de cumplir esta funcionalidad es, conceptualmente, bastante sencilla: la herramienta crea una base de datos en la que registra atributos de los ficheros y/o directorios de los que se quiere mantener un control de la integridad. Algunos de los atributos que se registran pueden ser: permisos, número de inodo, usuario propietario, grupo, tamaño del fichero, tamaño de crecimiento, etc. Además de almacenar estos atributos, la herramienta también crea un resumen de cada uno de los ficheros y/o directorios a controlar. Entre los algoritmos de resumen con los que puede operar

se encuentran: SHA-1, SHA-256, SHA-512 y MD5. Después de haber llevado a cabo estas operaciones, y de forma periódica, se vuelve a aplicar la función resumen sobre los ficheros y/o directorios controlados y el resultado es comparado con el almacenado en la base de datos. En caso de no coincidir ambos resúmenes, se notifica al administrador que se ha detectado algún cambio en los objetos controlados.

El contenido de la base de datos se puede tomar como una "fotografía" del estado normal del sistema, y este estado puede ser usado como referencia para detectar cambios sospechosos en dichos ficheros. Parece evidente que el control de integridad debe ser realizado sobre ficheros que no deben sufrir cambios a lo largo del tiempo, como por ejemplo, ficheros binarios del sistema, librerías y ficheros de cabecera. Por el contrario, no resulta lógico realizar un control de integridad sobre ficheros que suelen cambiar su contenido frecuentemente como pueden ser, por ejemplo, los registros (logs) del sistema.

2.2. Estudio de la situación actual

En esta sección se recoge el estudio realizado sobre los sistemas actualmente existentes que presenten características similares a las solicitadas para realizar este proyecto. Resulta importante realizar un estudio de estas características, ya que nos permite identificar a posibles competidores directos y qué servicios o características ofrecen, haciendo que destaquen del resto.

Por tanto, con el objetivo de conseguir que la herramienta final sea funcional, adecuada y ponga a disposición del usuario características más avanzadas que las demás herramientas, llevaremos a cabo un estudio de algunas de las herramientas que encontramos disponibles en la actualidad para la monitorización de la actividad y gestión de las principales herramientas de seguridad.

2.2.1. Aplicaciones web para la gestión de IPtables

En la actualidad podemos encontrar numerosas aplicaciones con interfaz gráfica para la gestión de IPtables[4]. Las más populares pueden ser Firestarter, Guarddog, KMyFirewall o fwbuilder. Sin embargo, todas ellas son empleadas como aplicaciones de escritorio. Esto supone dos claras desventajas frente a una aplicación web que lleve a cabo la misma funcionalidad:

1. El computador en el que se encuentre instalado el sistema de cortafuegos debe tener instalado un entorno gráfico para que sea posible acceder a este tipo de aplicaciones de escritorio. En este caso la ventaja de una aplicación web se encuentra en que el computador no necesita contar con un entorno gráfico.

2. El encargado de administrar el sistema de cortafuegos debe acceder directamente al computador en el que se encuentre instalado dicho sistema. En este caso la ventaja de una aplicación web se encuentra en que el encargado de administrar el sistema de cortafuegos puede acceder a la aplicación de gestión y monitorización desde cualquier otro computador de forma remota (siempre y cuando el propio sistema de cortafuegos permita las conexiones con dicho computador).

Teniendo en cuenta estas consideraciones, analizaremos dos aplicaciones web cuya funcionalidad puede ser similar a la funcionalidad que se desea lograr en la aplicación web que se desarrollará: phpIPtables y PHP Firewall Generator.

[4] Puesto que el objetivo de este trabajo es desarrollar una aplicación web con interfaz gráfica, no se tendrá en cuenta en este estudio aquellas aplicaciones con interfaces para consola (como pueden ser IPmenu o EasyTables).

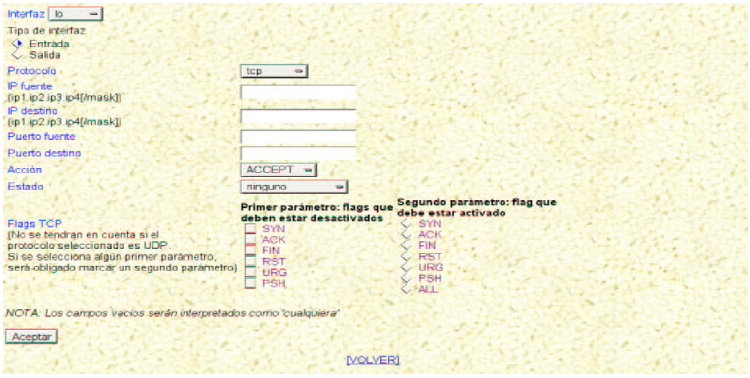

Figura 2.4.: Captura de pantalla de la aplicación phpIPtables.

phpIPtables

phpIPtables es una aplicación web para la gestión de IPtables cuyas principales características son las siguientes:

- La aplicación está desarrollada en lenguaje PHP.
- La aplicación sólo se encuentra disponible en castellano.
- Incluye la funcionalidad necesaria para la gestión de la aplicación IPtables en su versión 1.2.4.
- Incluye la funcionalidad necesaria para la gestión básica de las tablas FILTER y NAT.
- Incluye un fichero de configuración de la aplicación para establecer el valor de algunos parámetros necesarios para el correcto funcionamiento de la misma.
- No incluye la funcionalidad necesaria para lograr la persistencia de las reglas de cada una de las tablas. En caso de ser necesario asegurar la persistencia de las reglas, el usuario debe crear un script que sea ejecutado al iniciar el sistema.
- El desarrollo de ésta aplicación parece haberse estancado, puesto que la última versión disponible (versión 0.92 del año 2002) es todavía una versión en prueba.

Para obtener más información acerca de esta aplicación se puede consultar la página web del desarrollador, disponible en [22].

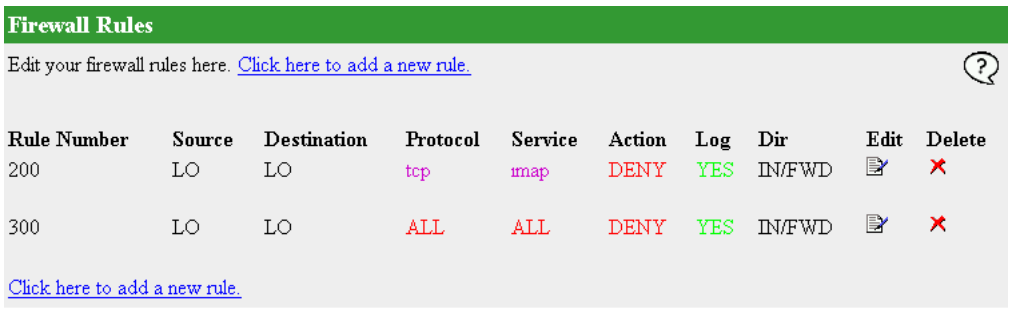

Figura 2.5.: Captura de pantalla de la aplicación PHP Firewall Generator.

PHP Firewall Generator

PHP Firewall Generator (versión 2.0) es una aplicación web para la gestión de IPtables cuyas principales características son las siguientes:

- La aplicación está desarrollada en lenguaje PHP.
- La aplicación está basada en el concepto de "objetos de red". De esta manera, el usuario final debe establecer todas las redes y computadores, lo que se conoce como "objetos de red", a los que, posteriormente, las reglas que definirá harán referencia.
- Incluye la funcionalidad necesaria para la generación de un script de configuración de IPtables.
- Incluye la funcionalidad necesaria para mostrar información de apoyo para la generación de reglas, como, por ejemplo, pueden ser la lista de protocolos almacenada en el fichero /etc/protocols, la lista de servicios almacenada en el fichero /etc/services o la lista de puertos abiertos obtenida a través del comando netfilter -a.
- No incluye la funcionalidad necesaria para actualizar el estado (añadir o borrar reglas, cambiar políticas por defecto de las tablas, ...) de la aplicación IPtables.
- No incluye la funcionalidad necesaria para lograr la persistencia de las reglas de cada una de las tablas.

Para obtener más información acerca de esta aplicación se puede consultar la página web del desarrollador, disponible en [27].

2.2.2. Aplicaciones web para la gestión de Snort

Al igual que en el caso de IPtables, en la actualidad es posible encontrar gran cantidad de aplicaciones con interfaz gráfica para la gestión de Snort. Las más populares pueden ser Easy IDS, Security Onion, Smooth-Sec, IDS Policy Manager, Snorby o las ya mencionadas ACID y BASE. Sin embargo, podemos encontrar las siguientes desventajas de estas aplicaciones frente a la aplicación web a desarrollar:

1. En el caso de Easy IDS, Security Onion y Smooth-Sec, no se trata de simples aplicaciones, sino que se trata de distribuciones del sistema operativo Linux que cuentan con las herramientas necesarias para llevar a cabo este trabajo.

2. La aplicación IDS Policy Manager, es empleada como una aplicación de escritorio, por lo que cuenta con las mismas desventajas expuestas en la sección 2.2.1.

3. Las aplicaciones Snorby, ACID y BASE pese a tratarse de aplicaciones web, sólo se encuentran orientadas a la monitorización y gestión de las alertas generadas por Snort.

Teniendo en cuenta estas consideraciones, analizaremos tres aplicaciones web cuya funcionalidad puede ser similar a la funcionalidad que se desea lograr en la aplicación web que se desarrollará: ACID, BASE y SNEZ. Estas aplicaciones pertenecerían a la tercera de las categorías establecidas anteriormente. Pese a que el objetivo que persiguen estas aplicaciones difiere en cierto modo del objetivo de la aplicación que buscamos desarrollar, nos encontramos en la obligación de analizar las características de éstas, puesto que no hemos encontrado disponible ninguna aplicación web que se haga cargo de la gestión de Snort (configuración de sus preprocesadores, reglas, variables, etc.).

Analysis Console for Intrusion Database (ACID)

En [8], sitio web mantenido por el autor de este software, se define el sistema Analysis Console for Intrusion Database (ACID) como un motor de búsqueda basado en el lenguaje PHP para buscar y procesar una base de datos de eventos de seguridad generados por sistemas IDS, cortafuegos u otros sistemas de monitorización de la red. Las características que aporta la herramienta son las siguientes:

- Permite construir consultas y provee una interfaz de búsqueda, cuya finalidad es encontrar alertas siguiendo distintos criterios de búsqueda, como pueden ser firmas, direcciones de origen o destino y puertos entre otros.

- Permite visualizar paquetes de forma que se puede mostrar de manera gráfica la información del paquete de red registrado en las alertas producidas.

- Permite llevar a cabo una gestión de las alertas producidas, la cual permite crear grupos de alertas relacionadas, eliminar o manejar falsos positivos, exportar las alertas de manera que puedan ser enviadas mediante correo electrónico o trasladar la información relativa a las alertas a otra base de datos de alertas.

- Permite generar diversos gráficos y estadísticas, basados en variables como el tiempo, firmas, protocolos, direcciones IP, ...

ACID da soporte para analizar una amplia variedad de eventos generados por distintas herramientas, entre los que encontramos: alertas generadas por Snort, registros binarios de TCPDUMP y mensajes almacenados en los registros (logs) del sistema producidos por ipchains, IPtables o ipfw.

Por tanto, parece bastante claro que el principal beneficio que supone el uso de esta herramienta es que aporta numerosas ayudas para realizar un análisis del tráfico inspeccionado.

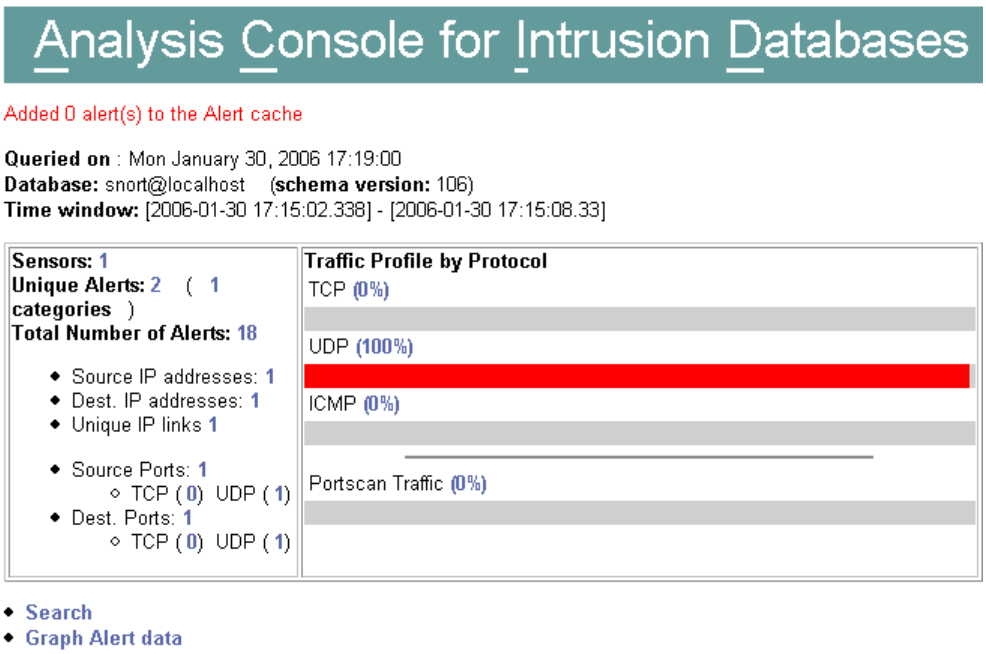

Figura 2.6.: Captura de pantalla de la aplicación ACID.

Basic Analisys and Security Engine (BASE)

Como veremos a continuación, este sistema, por la funcionalidad que ofrece, es muy similar a la herramienta presentada anteriormente.

En [6] se define el sistema Basic Analysis and Security Engine (BASE) como un sistema basado en la herramienta ACID que describimos anteriormente. En términos generales esta herramienta provee una interfaz web para realizar consultas y analizar las alertas producidas por Snort.

La herramienta es una interfaz web para realizar análisis de las intrusiones que Snort detecta en la red en la que trabaja. Cuenta con un sistema basado en usuarios (los cuales poseen un rol en el sistema y deben autenticarse en el mismo). De esta forma el administrador puede decidir qué y cuánta cantidad de información puede visualizar cada uno de los usuarios. Además su uso es bastante sencillo ya que dispone de un sistema de configuración basado en interfaces web lo cual evita a los usuarios tener que manipular ficheros de configuración.

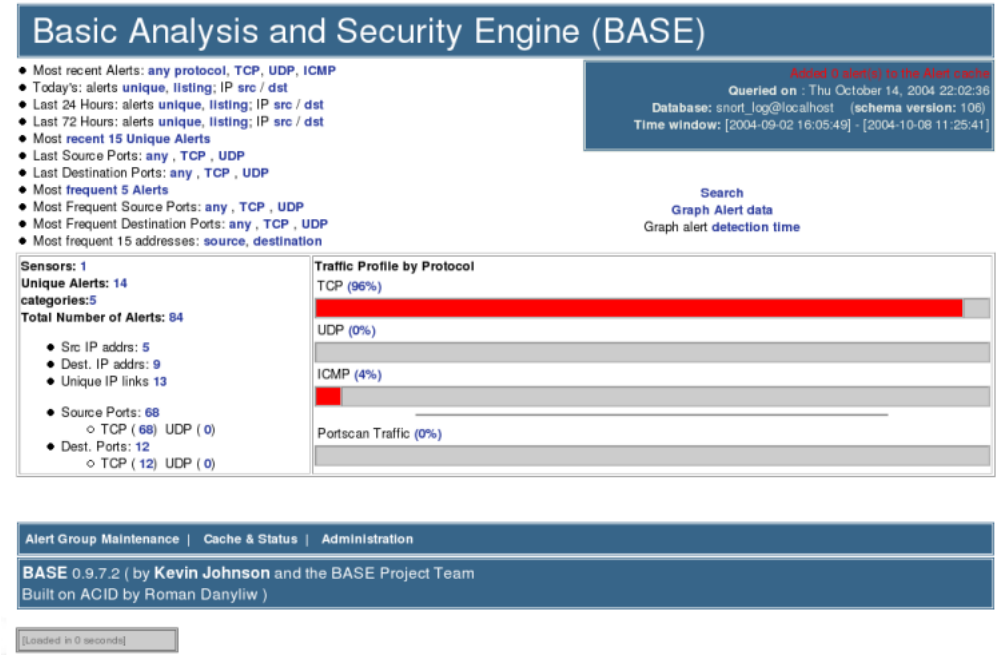

Figura 2.7.: Captura de pantalla de la aplicación BASE.

SNEZ

SNEZ es una aplicación web para la gestión y visualización de alertas y paquetes registrados por Snort, cuyas principales características son las siguientes:

- La aplicación está desarrollada en lenguaje PHP.
- Incluye la funcionalidad necesaria tanto para la visualización como para la gestión de las alertas producidas por Snort (filtrar las mismas por distintos parámetros como, por ejemplo, dirección IP de origen o fecha).

- Incluye la funcionalidad necesaria para la visualización en tiempo real de los paquetes inspeccionados por Snort.
- Incluye la funcionalidad necesaria para la visualización del estado de los procesos relacionados con Snort.
- Incluye la funcionalidad necesaria para que varios usuarios con distintos roles puedan hacer uso de la aplicación.
- Incluye un fichero de configuración de la aplicación para establecer el valor de algunos parámetros necesarios para el correcto funcionamiento de la misma.

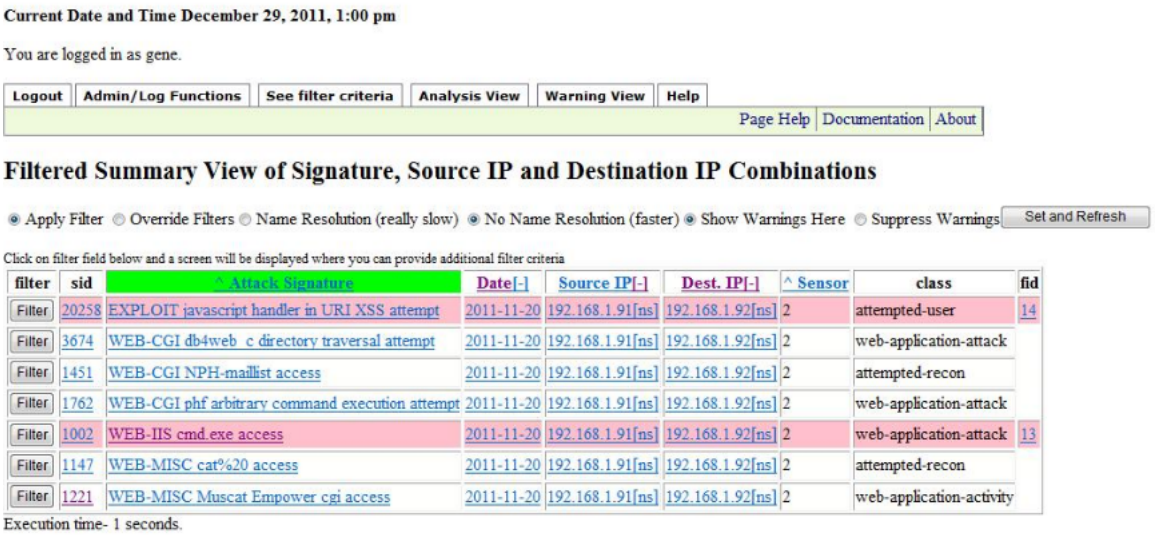

Figura 2.8.: Captura de pantalla de la aplicación SNEZ.

Para obtener más información acerca de esta aplicación se puede consultar la página web del desarrollador, disponible en [14].

2.3. Valoración de la situación actual

Tras realizar el estudio de la situación actual, en cuanto a aplicaciones similares a la que se desea desarrollar, podemos darnos cuenta de que existe un nicho de mercado en cuanto a aplicaciones web cuya función sea la de monitorizar la actividad y gestionar las principales herramientas para la defensa de sistemas informáticos como son un cortafuegos y un sistema de detección de intrusiones. Esto se debe, principalmente, a que la mayoría de aplicaciones con esta funcionalidad:

1. Están orientadas a ser aplicaciones de escritorio.
2. Están orientadas únicamente a la monitorización de la actividad del sistema, dejando de lado la gestión de las aplicaciones.
3. Su fase de desarrollo ha sido interrumpida y/o no ofrecen las actualizaciones necesarias para el correcto funcionamiento de las mismas.

Por otra parte, para lograr posicionar la aplicación web que se desea desarrollar por delante de las aplicaciones anteriormente analizadas, se deberán incluir en ella las características más beneficiosas de cada una de ellas, así como nuevas características que aumenten o mejoren la funcionalidad y seguridad de la misma. Por ejemplo, algunas de las características que solventarían las desventajas enumeradas antes serán:

1. Desarrollar una aplicación accesible desde cualquier computador que disponga de acceso a Internet.
2. Desarrollar una aplicación que permita tanto monitorizar la actividad como configurar las herramientas de seguridad.
3. Desarrollar una aplicación que permita la gestión de varias herramientas de seguridad.
4. Desarrollar una aplicación de forma modular, de manera que su mantenimiento y actualización sean sencillas de realizar.

Capítulo 3.

Plan de trabajo y presupuesto

3.1. Plan de trabajo

En esta sección se llevará a cabo una planificación del trabajo que es necesario realizar para controlar el progreso del proyecto. En la página siguiente se muestra el diagrama de Gantt que recoge la planificación inicial del proyecto.

Es necesario destacar, que pese a que según esta planificación el proyecto debería haber sido completado antes del primer plazo de entrega del mismo, durante el transcurso de su desarrollo aparecieron, principalmente, los siguientes inconvenientes:

- Se dio mayor prioridad a los trabajos del resto de asignaturas pendientes por superar que al trabajo a realizar para este proyecto. Esto supuso que el tiempo diario dedicado al desarrollo del proyecto se viera reducido, llegando a ser, en algunas ocasiones, muy inferior a las cinco horas estimadas inicialmente.

- Se dio mayor prioridad al estudio de los exámenes finales de las asignaturas, entonces no superadas, del mes de mayo que al trabajo a realizar para este proyecto. Esto, al igual que en el caso anterior, supuso que el tiempo diario dedicado al desarrollo del proyecto se viera muy reducido.

- La estimación del tiempo necesario para completar algunas de las tareas planificadas fue poco realista, sobretodo en las fases de preparación del entorno (debido a la necesidad de tener que compilar e instalar software de forma manual) e implementación (debido a la necesidad de implementar componentes con los que nunca nos habíamos enfrentado), en las que el tiempo real fue algo más que el tiempo que se estimó.

Debido a estos inconvenientes el proceso de desarrollo sufrió graves retrasos, los cuales fueron más que suficientes como para que resultara necesario tener que prolongar el tiempo de trabajo durante tres meses más de lo previsto.

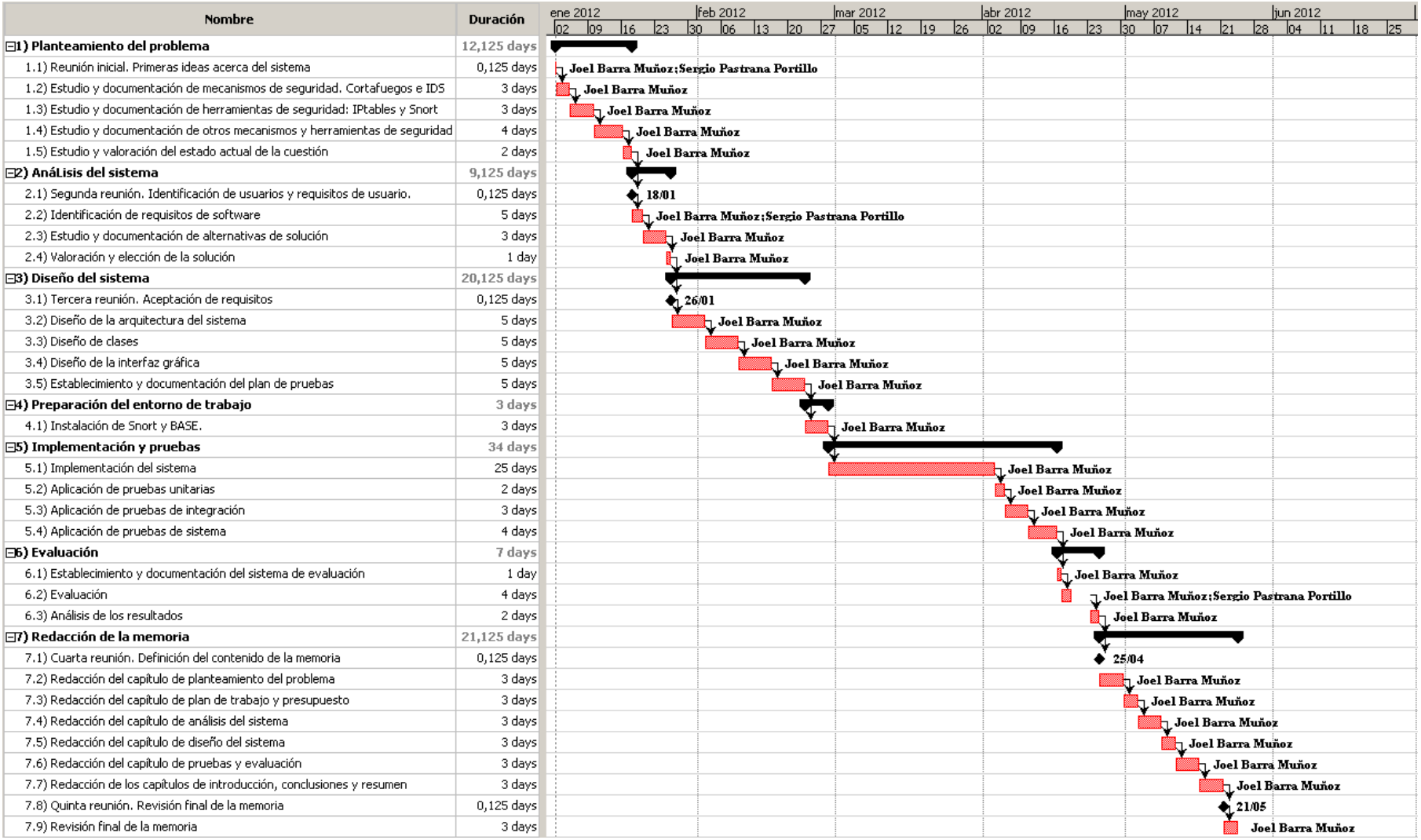

Figura 3.1.: Diagrama de Gantt para la planificación inicial del trabajo.

3.2. Presupuesto

En esta sección se va a intentar llevar a cabo una aproximación del coste que supone llevar a cabo este trabajo. Los costes de personal han sido estimados haciendo uso del total de horas necesarias para completar el trabajo que fueron planificadas en el apartado anterior. El desglose de todos los costes del proyecto se muestra en las siguientes tablas.

Tabla 3.1.: Costes aproximados de personal.

PERSONAL					
Apellidos y nombre	*Categoría*	*Estimación*	*Dedicación*	*Coste unitario*	*Coste*
Barra Muñoz, Joel	Ingeniero	106.5 días	5 horas/día	14 €/hora	7455 €
				Total	7455 €

Tabla 3.2.: Costes aproximados de equipos.

EQUIPOS			
Cantidad	*Concepto*	*Coste unitario*	*Coste*
1	Ordenador portátil Asus K53SV	650 €	650 €
1	Disco duro externo Western Digital 500 GB	89 €	89 €
		Total	739 €

Tabla 3.3.: Costes aproximados de software.

SOFTWARE			
Cantidad	*Concepto*	*Coste unitario*	*Coste*
1	VMWare Player 3.1.4	0 €	0 €
1	Linux Ubuntu 10.10	0 €	0 €
1	Rational Software Architect/Eclipse Indigo	0 €	0 €
1	Snort 2.9.2	0 €	0 €
1	BASE	0 €	0 €
1	IPtables 1.4.4	0 €	0 €
1	MySQL Server	0 €	0 €
1	MySQL Client	0 €	0 €
1	Apache Tomcat 6.0.35	0 €	0 €
1	Procesador de texto LYX 2.0.3	0 €	0 €
1	JabRef 2.7	0 €	0 €
1	OpenProj 1.4	0 €	0 €
		Total	0 €

Tabla 3.4.: Resumen de todos los costes aproximados del proyecto.

Concepto	**Coste**
Personal	7455 €
Equipos	739 €
Software	0 €
Subtotal	8194 €
Costes indirectos	10 %
Impuestos	21 %
Total	10906.21 €

Capítulo 4.

Análisis del sistema

4.1. Identificación del perfil de usuario final

A la hora de realizar el análisis del sistema, uno de los aspectos más relevantes que se deben conocer es a quién va destinada la aplicación. En esta sección desarrollaremos un breve perfil de los posibles usuarios finales del sistema.

Las capacidades con las que se supone que cuentan los usuarios interesados en la aplicación a desarrollar serán las siguientes:

- Usuario frecuente y con amplio conocimiento en el manejo y administración del sistema operativo Linux, en cualquiera de sus distribuciones.
- Usuario con amplia experiencia en el ámbito de la seguridad de sistemas de información.
- Usuario frecuente y con amplio conocimiento en el manejo de sistemas cortafuegos. No es necesario que el usuario cuente con mucha experiencia en el uso de la herramienta IPtables, aunque sí deberá disponer de unas nociones básicas de la misma.
- Usuario frecuente y con amplio conocimiento en el manejo de sistemas de detección de intrusiones. No es necesario que el usuario cuente con mucha experiencia en el uso de la herramienta Snort, aunque sí deberá disponer de unas nociones básicas de la misma.
- Usuario frecuente y con amplio conocimiento en el manejo de navegadores web, en concreto del navegador Mozilla Firefox.

Para lograr un sistema usable y útil para el usuario, el posterior diseño del sistema deberá realizarse teniendo en mente este perfil de usuario.

4.2. Requisitos de usuario

En esta sección se recogen los requisitos de usuario que el sistema a desarrollar deberá cumplir. Los requisitos que aquí se recogen no son definitivos, es decir, podrán ser modificados a lo largo de la fase de desarrollo de la aplicación para garantizar que la funcionalidad del producto final sea correcta y completa.

Antes de comenzar con la identificación de los requisitos, plantearemos algunas cuestiones que resulta conveniente aclarar antes de comenzar a realizar dicho trabajo. A la hora de identificar los requisitos de usuario del sistema a desarrollar, dispondremos de dos tipos de requisitos:

- Requisitos de capacidad. Este tipo de requisitos recogerán las necesidades del usuario para lograr su objetivo.
- Requisitos de restricción. Este tipo de requisitos recogerán las restricciones impuestas por el usuario para lograr alcanzar su objetivo.

A su vez, para los requisitos de capacidad, con el fin de facilitar el análisis de cada uno de ellos, dispondremos de las siguientes categorías:

- Requisitos funcionales. Los requisitos clasificados en esta categoría están referidos al problema que debe resolver el sistema a desarrollar.
- Requisitos de rendimiento. Los requisitos clasificados en esta categoría están referidos a aspectos como tiempo de respuesta de la aplicación, capacidad de disco, necesidades de memoria, ...
- Requisitos de seguridad. Los requisitos clasificados en esta categoría están referidos a aspectos de seguridad que requiere el sistema a desarrollar.
- Requisitos de implantación. Los requisitos clasificados en esta categoría están referidos al entorno en el que el sistema a desarrollar debe trabajar.
- Requisitos de disponibilidad del sistema. Los requisitos clasificados en esta categoría están referidos al tiempo que el sistema a desarrollar debe encontrarse operativo.

Por último, cada requisito dispondrá de los siguientes atributos:

- Identificador. Cada requisito dispondrá de un identificador único que lo represente de manera unívoca. Este atributo contará con el siguiente formato:
  - CAP_XXX si el requisito en cuestión es de capacidad. Será necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar el requisito.
  - RES_XXX si el requisito en cuestión es de restricción. Será necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar el requisito.

- Categoría. Este atributo identificará, en el caso de los requisitos de capacidad, a qué categoría de las anteriormente mencionadas pertenece el requisito. Este atributo, por tanto, podrá tomar los siguientes valores:
  - Funcional.
  - De rendimiento.
  - De seguridad.
  - De implantación.
  - De disponibilidad del sistema.
- Prioridad. Este atributo identificará la prioridad con la que el requisito debe ser resuelto. Este atributo podrá tomar los siguientes valores:
  - Alta. La consecución del requisito se llevará a cabo antes que cualquier otro requisito.
  - Media. La consecución del requisito sólo se llevará a cabo tras haberse cumplido todos los requisitos de prioridad alta.
  - Baja. La consecución del requisito sólo se llevará a cabo tras haberse cumplido todos los requisitos de prioridad media y alta.
- Estabilidad. Este atributo identificará la probabilidad de que el requisito varíe, ya sea modificado o eliminado, a lo largo del ciclo de vida del proyecto. Este atributo podrá tomar los siguientes valores:
  - Estable. La probabilidad de que el requisito sea modificado es baja o prácticamente nula.
  - Inestable. La probabilidad de que el requisito sea modificado es media o alta.
- Verificabilidad. Este atributo identificará la posibilidad de que se compruebe que el sistema, una vez desarrollado, cumple con el requisito en cuestión. Este atributo podrá tomar los siguientes valores:
  - Alta. La posibilidad de verificar que el requisito se cumple es alta.
  - Media. La posibilidad de verificar que el requisito se cumple es media.
  - Baja. La posibilidad de verificar que el requisito se cumple es baja.
- Descripción. Este atributo identificará cuál es el significado del requisito, expresando el mismo de una forma clara, sencilla, coherente, completa, concisa y, sobre todo, sin ambigüedades.

A continuación, se muestra un ejemplo de la manera en la que serán documentados los requisitos de usuario del sistema.

Tabla 4.1.: Ejemplo de documentación de un requisito de usuario.

CAP_000	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	Los requisitos de usuario deben documentarse siguiendo este formato.

4.2.1. Identificación de requisitos de usuario

Requisitos de capacidad

Tabla 4.2.: Requisito CAP_001.

CAP_001	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de configurar los parámetros necesarios para su funcionamiento.

Tabla 4.3.: Requisito CAP_002.

CAP_002	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe incluir una sección dedicada a información de ayuda para los usuarios de la aplicación.

Tabla 4.4.: Requisito CAP_003.

CAP_003	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer de forma diferenciada la funcionalidad para Snort y para IPtables.

Tabla 4.5.: Requisito CAP_004.

CAP_004	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de monitorizar la actividad desarrollada por la aplicación Snort.

Tabla 4.6.: Requisito CAP_005.

CAP_005	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar los parámetros de Snort.

Tabla 4.7.: Requisito CAP_006.

CAP_006	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer información de apoyo acerca de cuáles son los posibles valores que se pueden aplicar en cada parámetro configurable de la aplicación Snort.

Tabla 4.8.: Requisito CAP_007.

CAP_007	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de visualizar las reglas de detección que Snort aplica.

Tabla 4.9.: Requisito CAP_008.

CAP_008	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe mostrar si la aplicación Snort se encuentra en ejecución o parada.

Tabla 4.10.: Requisito CAP_009.

CAP_009	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar la ejecución de la aplicación Snort.

Tabla 4.11.: Requisito CAP_010.

CAP_010	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar todas las cadenas de todas las tablas que maneja IPtables.

Tabla 4.12.: Requisito CAP_011.

CAP_011	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar las reglas de todas las cadenas que maneja IPtables.

Tabla 4.13.: Requisito CAP_012.

CAP_012	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de visualizar las reglas de la aplicación IPtables.

Tabla 4.14.: Requisito CAP_013.

CAP_013	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer información de apoyo acerca de cuáles son los posibles valores que se pueden aplicar en cada parámetro configurable de la aplicación IPtables.

Tabla 4.15.: Requisito CAP_014.

CAP_014	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de guardar el estado actual de la aplicación IPtables.

Tabla 4.16.: Requisito CAP_015.

CAP_015	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de restaurar un estado, previamente guardado, de la aplicación IPtables.

Tabla 4.17.: Requisito CAP_016.

CAP_016	
Categoría	De rendimiento.
Prioridad	Baja.
Estabilidad	Estable.
Verificabilidad	Baja.
Descripción	El sistema debe tardar menos de 30 segundos en ofrecer una respuesta ante una petición realizada.

Tabla 4.18.: Requisito CAP_017.

CAP_017	
Categoría	De rendimiento.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Media.
Descripción	El sistema debe mantener constantemente actualizado el estado de las aplicaciones gestionadas.

Tabla 4.19.: Requisito CAP_018.

CAP_018	
Categoría	De seguridad.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Media.
Descripción	El sistema debe ocultar las opciones que el usuario no debe configurar.

Tabla 4.20.: Requisito CAP_019.

CAP_019	
Categoría	De seguridad.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema no debe actualizar el estado de las aplicaciones gestionadas en caso de que el usuario introduzca un valor in-correcto para algún parámetro.

Tabla 4.21.: Requisito CAP_020.

CAP_020	
Categoría	De seguridad.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Baja.
Descripción	El sistema debe, en caso de que el usuario cometa un error, proporcionar información para la identificación y corrección del mismo.

Tabla 4.22.: Requisito CAP_021.

CAP_021	
Categoría	De implantación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ejecutar en cualquier computador que cuente con un sistema operativo Ubuntu 10.10 o superior.

Tabla 4.23.: Requisito CAP_022.

CAP_022	
Categoría	De implantación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe permitir monitorizar la actividad y gestionar la aplicación Snort versión 2.9.2.

Tabla 4.24.: Requisito CAP_023.

CAP_023	
Categoría	De implantación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe permitir monitorizar la actividad y gestionar la aplicación IPtables versión 1.4.4.

Tabla 4.25.: Requisito CAP_024.

CAP_024	
Categoría	De implantación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ejecutar en un servidor web Apache Tomcat versión 6.0.35 o superior.

Tabla 4.26.: Requisito CAP_025.

CAP_025	
Categoría	De implantación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ser accesible a través del navegador web Mozilla Firefox versión 14.0.1 o superior.

Requisitos de restricción

Tabla 4.27.: Requisito RES_001.

RES_001	
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema a desarrollar debe tratarse de una aplicación web.

Tabla 4.28.: Requisito RES_002.

RES_002	
Prioridad	Alta.
Estabilidad	Alta.
Verificabilidad	Alta.
Descripción	La información proporcionada por el sistema debe estar expresada en inglés.

Tabla 4.29.: Requisito RES_003.

RES_003	
Prioridad	Alta.
Estabilidad	Alta.
Verificabilidad	Alta.
Descripción	La visualización de las alertas generadas por Snort será realizada a través de la aplicación BASE.

4.3. Requisitos de software

En esta sección se recogen los requisitos de software que el sistema a desarrollar deberá cumplir. El objetivo de estos requisitos de software es analizar los requisitos de usuario establecidos y producir un conjunto de requisitos que debe cumplir el sistema, que sea tan completo y correcto como sea posible.

El desarrollo de esta tarea de identificación de requisitos de software se lleva a cabo analizando los requisitos de usuario y construyendo una descripción coherente y completa de lo que el producto final debe hacer.

4.3.1. Casos de uso

Para lograr el objetivo planteado anteriormente, se utilizará la técnica de los casos de uso. Con los casos de uso se logrará especificar aquello que debe hacer el sistema desde el punto de vista del usuario. Es decir, esta técnica describe el uso del sistema y como éste interactúa tanto con el usuario como con otros componentes del mismo.

Al llevar a cabo este análisis haremos uso, en primer lugar, de una descripción gráfica de los casos de uso, para después realizar una descripción textual de cada uno de los casos de uso planteados.

La descripción textual mencionada anteriormente debe realizarse de manera concisa y descriptiva, ayudándonos para su especificación de los siguientes atributos:

- Identificador. Cada caso de uso dispondrá de un identificador único que lo represente de manera unívoca. Este identificador será CU-XXX, resultando necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar el caso de uso.
- Actores. Este atributo identificará los posibles roles de usuario que pueden intervenir en el caso de uso.
- Objetivo. Este atributo definirá brevemente la finalidad que persigue el actor al realizar el caso de uso en cuestión.
- Descripción. Este atributo definirá brevemente el proceso llevado a cabo en el caso de uso.
- Precondiciones. Este atributo identificará las condiciones que deben cumplirse para que el caso de uso en cuestión pueda llevarse a cabo.
- Postcondiciones. Este atributo identificará las condiciones que deben cumplirse tras haberse llevado a cabo el caso de uso en cuestión.
- Escenario básico. Este atributo definirá la ejecución que compone el caso de uso en cuestión.
- Escenario alternativo. Este atributo definirá la ejecución con condiciones de error (o caminos de decisión) distintos al básico.

A continuación, se muestra un ejemplo de la manera en la que serán documentados los casos de uso.

Tabla 4.30.: Ejemplo de documentación de un caso de uso.

CU_000	
Actores	Administrador de seguridad.
Objetivo	Iniciar la aplicación.
Descripción	El actor inicia la ejecución del navegador web e introduce la dirección URL.
Precondiciones	1. El navegador web Mozilla Firefox (versión 14.0.1) se encuentra instalado en el computador. 2. Se tiene conexión con el servidor a través de Internet o una red local.
Postcondiciones	El navegador muestra la página de inicio de la aplicación.
Escenario básico	1. Abrir el navegador web. 2. Introducir la dirección URL en la barra de dirección del navegador web.
Escenario alternativo	-

4.3.2. Identificación de actores

A continuación describimos los roles participantes en la interacción con el sistema:

- Administrador de seguridad (o simplemente "Administrador"). Este rol será interpretado por aquella persona, o personas, encargada de gestionar la seguridad de un sistema Linux en el que se encuentre instalado el sistema que se va a desarrollar. El perfil de las personas que interpreten este rol fue expuesto en la sección 4.1.

- Software de apoyo a la monitorización y la gestión (o simplemente "Software de apoyo"). Este rol será interpretado por aquel software que es empleado para la gestión de las aplicaciones Snort e IPtables.

4.3.3. Identificación de casos de uso

A continuación se muestra una descripción gráfica de los casos de uso, en los que quedan identificados actores, el sistema y los casos de uso del mismo.

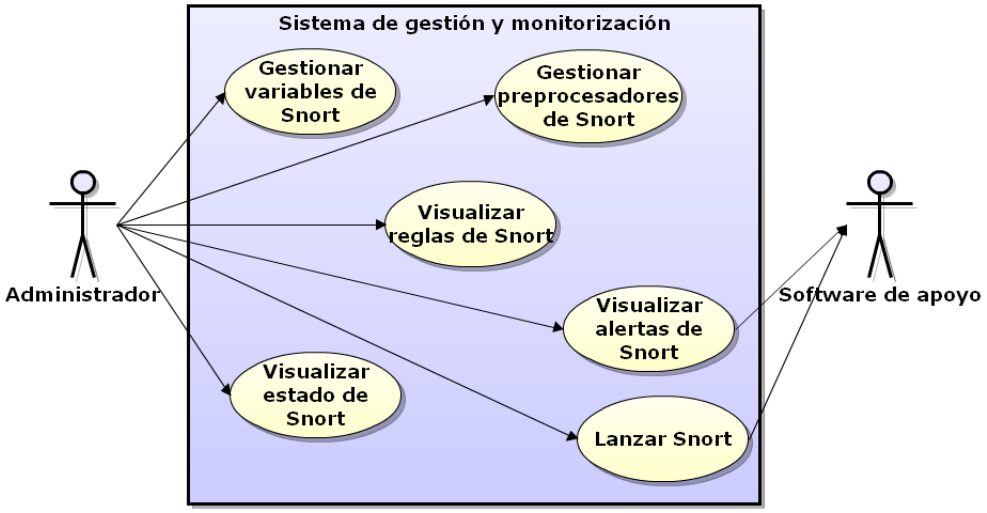

Figura 4.1.: Diagrama de casos de uso (Parte I).

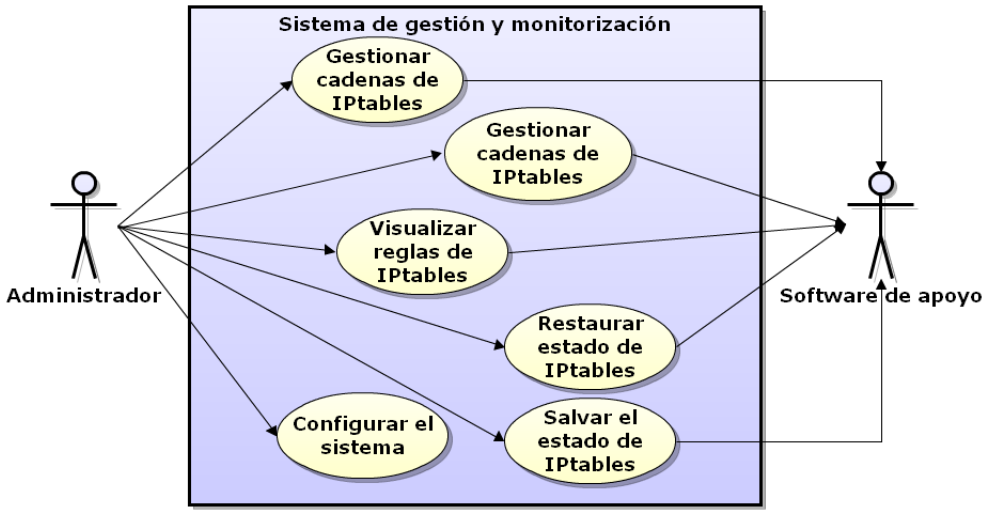

Figura 4.2.: Diagrama de casos de uso (Parte II).

A continuación se describirán de manera más formal cada uno de los casos de uso, sirviendo como ayuda los diagramas ofrecidos anteriormente. Cada caso de uso será documentado siguiendo el formato explicado al comienzo de esta sección.

Tabla 4.31.: Caso de uso CU_001.

CU_001	
Actores	Administrador de seguridad.
Objetivo	Gestionar las variables de Snort.
Descripción	El administrador desea gestionar las variables de configuración de Snort.
Precondiciones	El sistema ha sido configurado indicando la ruta en la que se encuentra el fichero de configuración de Snort.
Postcondiciones	Los cambios realizados en las variables quedan registrados en el fichero de configuración de Snort.
Escenario básico	1. El administrador accede a la sección de Snort. 2. El administrador selecciona el tipo de variable a gestionar. 3. El administrador realiza los cambios deseados. 4. El administrador pulsa el botón "Save" para hacer los cambios permanentes en el sistema. 5. El sistema lleva a cabo los cambios necesarios en el fichero de configuración de Snort.
Escenario alternativo 1	4a. El sistema detecta que algunos de los valores introducidos no son válidos. 5. El sistema señala qué valores son incorrectos. 6. El administrador corrige los valores incorrectos. 7. Volver al paso 4.
Escenario alternativo 2	4a. El sistema no encuentra el fichero de configuración de Snort en la ruta especificada en la configuración del sistema. 5. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.
Escenario alternativo 3	4a. El sistema no puede leer o escribir el fichero de configuración de Snort. 5. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.32.: Caso de uso CU_002.

CU_002	
Actores	Administrador de seguridad.
Objetivo	Gestionar los preprocesadores de Snort.
Descripción	El administrador desea gestionar los preprocesadores de Snort.
Precondiciones	-
Postcondiciones	Los cambios realizados en los preprocesadores quedan registrados en el fichero de configuración de Snort.
Escenario básico	1. El administrador accede a la sección de Snort. 2. El administrador selecciona el preprocesador a gestionar. 3. El administrador realiza los cambios deseados. 4. El administrador pulsa el botón "Save" para hacer los cambios permanentes en el sistema. 5. El sistema lleva a cabo los cambios necesarios en el fichero de configuración de Snort.
Escenario alternativo 1	4a. El sistema detecta que algunos de los valores introducidos no son válidos. 5. El sistema señala qué valores son incorrectos. 6. El administrador corrige los valores incorrectos. 7. Volver al paso 4.
Escenario alternativo 2	4a. El sistema no encuentra el fichero de configuración de Snort en la ruta especificada en la configuración del sistema. 5. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.
Escenario alternativo 3	4a. El sistema no puede leer o escribir el fichero de configuración de Snort. 5. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.33.: Caso de uso CU_003.

CU_003	
Actores	Administrador de seguridad.
Objetivo	Visualizar las reglas de Snort.
Descripción	El administrador desea visualizar las reglas de detección de Snort.
Precondiciones	Deben existir reglas definidas en la aplicación Snort.
Postcondiciones	-
Escenario básico	1. El administrador accede a la sección de Snort. 2. El administrador selecciona el fichero de reglas a visualizar. 3. El sistema muestra las reglas de detección de dicho fichero . que Snort está aplicando
Escenario alternativo	4a. El sistema no puede leer el fichero de reglas. 5. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.34.: Caso de uso CU_004.

CU_004	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Visualizar las alertas de Snort.
Descripción	El administrador desea visualizar las alertas generadas por Snort.
Precondiciones	La aplicación BASE se encuentra instalada en el sistema.
Postcondiciones	1. Las alertas mostradas son todas aquellas que Snort ha generado durante el desarrollo de su actividad. 2. El sistema permitirá manejar completamente la aplicación BASE.
Escenario básico	1. El administrador accede a la sección de Snort. 2. El administrador selecciona acceder a la aplicación BASE. 3. El sistema ejecuta la aplicación BASE mostrando su interfaz al usuario.
Escenario alternativo	-

Tabla 4.35.: Caso de uso CU_005.

CU_005	
Actores	Administrador de seguridad.
Objetivo	Visualizar el estado de Snort.
Descripción	El administrador desea conocer si Snort se encuentra en ejecución o no.
Precondiciones	-
Postcondiciones	-
Escenario básico	1. El administrador accede a la sección de Snort. 2. El sistema muestra un mensaje indicando si la aplicación se encuentra en ejecución o no.
Escenario alternativo	3a. El sistema no puede lanzar la ejecución de Snort. 4. El sistema muestra un mensaje indicando que la aplicación no se encuentra en ejecución.

Tabla 4.36.: Caso de uso CU_006.

CU_006	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Lanzar Snort.
Descripción	El administrador desea lanzar la ejecución de Snort.
Precondiciones	-
Postcondiciones	La aplicación Snort se encuentra ejecutando en el sistema.
Escenario básico	1. El administrador accede a la sección de Snort. 2. El administrador selecciona lanzar la ejecución de Snort 3. El sistema emplea el software de apoyo para llevar a cabo la ejecución de Snort. 4. El sistema muestra un mensaje indicando que la aplicación se encuentra en ejecución.
Escenario alternativo	3a. El sistema no puede lanzar la ejecución de Snort. 4. El sistema muestra un mensaje indicando que la aplicación no se encuentra en ejecución.

Tabla 4.37.: Caso de uso CU_007.

CU_007	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Gestionar cadenas de IPtables.
Descripción	El administrador desea gestionar las cadenas de las tablas ofrecidas por la aplicación IPtables.
Precondiciones	-
Postcondiciones	Los cambios realizados en las cadenas quedan registrados en la aplicación IPtables.
Escenario básico	1. El administrador accede a la sección de IPtables. 2. El administrador selecciona la tabla de la que desea gestionar las cadenas. 3. El administrador realiza los cambios deseados. 4. El administrador pulsa el botón "Save" para hacer los cambios permanentes en el sistema. 5. El sistema emplea el software de apoyo para llevar a cabo la configuración deseada por el administrador.
Escenario alternativo 1	4a. El sistema detecta que algunos de los valores introducidos no son válidos. 5. El sistema señala qué valores son incorrectos. 6. El administrador corrige los valores incorrectos. 7. Volver al paso 4.
Escenario alternativo 2	5a. El sistema no puede aplicar los cambios especificados. 6. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.38.: Caso de uso CU_008.

CU_008	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Gestionar reglas de IPtables.
Descripción	El administrador desea gestionar las reglas de las cadenas contenidas en la aplicación IPtables.
Precondiciones	-
Postcondiciones	Los cambios realizados en las reglas quedan registrados en la aplicación IPtables.
Escenario básico	1. El administrador accede a la sección de IPtables. 2. El administrador selecciona la tabla de la que desea gestionar las reglas. 3. El administrador realiza los cambios deseados. 4. El administrador pulsa el botón "Save" para hacer los cambios permanentes en el sistema. 5. El sistema emplea el software de apoyo para llevar a cabo la configuración deseada por el administrador.
Escenario alternativo 1	4a. El sistema detecta que algunos de los valores introducidos no son válidos. 5. El sistema señala qué valores son incorrectos. 6. El administrador corrige los valores incorrectos. 7. Volver al paso 4.
Escenario alternativo 2	5a. El sistema no puede aplicar los cambios especificados. 6. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.39.: Caso de uso CU_009.

CU_009	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Visualizar reglas de IPtables.
Descripción	El administrador desea visualizar las reglas de las cadenas contenidas en la aplicación IPtables.
Precondiciones	-
Postcondiciones	-
Escenario básico	1. El administrador accede a la sección de IPtables. 2. El administrador selecciona la tabla de la que desea visualizar las reglas. 3. El sistema emplea el software de apoyo para recopilar las reglas contenidas en la tabla seleccionada. 4. El sistema muestra las reglas contenidas en la tabla seleccionada.
Escenario alternativo	3a. El sistema no puede acceder a las reglas de IPtables. 4. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.40.: Caso de uso CU_010.

CU_010	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Guardar el estado de IPtables.
Descripción	El administrador desea guardar el estado de la aplicación IPtables.
Precondiciones	-
Postcondiciones	El estado de la aplicación queda almacenado para una posterior restauración del mismo.
Escenario básico	1. El administrador accede a la sección de IPtables. 2. El administrador selecciona la opción de guardar el estado de IPtables. 3. El sistema emplea el software de apoyo para guardar el estado de IPtables. 4. El sistema muestra la fecha y hora en la que fue guardado por última vez el estado de la aplicación.
Escenario alternativo	3a. El sistema no puede guardar el estado de IPtables. 4. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.

Tabla 4.41.: Caso de uso CU_011.

CU_011	
Actores	Administrador de seguridad, Software de apoyo a la monitorización y la gestión.
Objetivo	Restaurar el estado de IPtables.
Descripción	El administrador desea restaurar un estado de la aplicación IPtables.
Precondiciones	El estado a restaurar ha sido guardado previamente.
Postcondiciones	El estado de la aplicación tras la restauración corresponde con el estado guardado de la aplicación.
Escenario básico	1. El administrador accede a la sección de IPtables. 2. El administrador selecciona la opción de restaurar el estado de IPtables. 3. El sistema muestra las características del estado que va a ser restaurado. 4. El sistema emplea el software de apoyo para restaurar el estado de IPtables.
Escenario alternativo 1	4a. El sistema no puede restaurar el estado de IPtables. 5. El sistema muestra una pantalla de error especificando el código, la descripción y los posibles motivos del error.
Escenario alternativo 2	2a. El sistema no encuentra ningún estado guardado previamente. 3. El sistema muestra un mensaje informando al administrador de que el estado debe ser guardado previamente.

Tabla 4.42.: Caso de uso CU_012.

CU_012	
Actores	Administrador de seguridad.
Objetivo	Configurar el sistema.
Descripción	El administrador desea configurar algún parámetro del sistema.
Precondiciones	-
Postcondiciones	-
Escenario básico	1. El administrador accede al fichero de configuración de la aplicación. 2. El administrador configura los parámetros deseados.
Escenario alternativo	-

4.3.4. Requisitos de software

Tras el análisis de los casos de uso, identificaremos los requisitos software para el sistema a desarrollar.

Para la especificación de estos requisitos haremos uso de los siguientes atributos:

- Identificador. Cada requisito dispondrá de un identificador único que lo represente de manera unívoca. Este atributo será RS_XXX, resultando necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar el caso de uso.
- Categoría. Este atributo identificará a qué categoría pertenece el requisito. Este atributo podrá tomar los siguientes valores:
  - Funcional.
  - De rendimiento.
  - De interfaz.
  - De operación.
  - De recursos.
  - De comprobación.
  - De aceptación de pruebas.
  - De documentación.
  - De seguridad.
  - De mantenimiento.
  - De calidad.
- Prioridad. Este atributo identificará la prioridad con la que el requisito debe ser resuelto. Este atributo podrá tomar los siguientes valores:
  - Alta. La consecución del requisito se llevará a cabo antes que cualquier otro requisito.
  - Media. La consecución del requisito sólo se llevará a cabo tras haberse cumplido todos los requisitos de prioridad alta.
  - Baja. La consecución del requisito sólo se llevará a cabo tras haberse cumplido todos los requisitos de prioridad media y alta.
- Estabilidad. Este atributo identificará la probabilidad de que el requisito varíe, ya sea modificado o eliminado, a lo largo del ciclo de vida del proyecto. Este atributo podrá tomar los siguientes valores:
  - Estable. La probabilidad de que el requisito sea modificado es baja o prácticamente nula.
  - Inestable. La probabilidad de que el requisito sea modificado es media o alta.

- Verificabilidad. Este atributo identificará la posibilidad de que se compruebe que el sistema, una vez desarrollado, cumple con el requisito en cuestión. Este atributo podrá tomar los siguientes valores:
  - Alta. La posibilidad de verificar que el requisito se cumple es alta.
  - Media. La posibilidad de verificar que el requisito se cumple es media.
  - Baja. La posibilidad de verificar que el requisito se cumple es baja.
- Descripción. Este atributo identificará cuál es el significado del requisito, expresando el mismo de una forma clara, sencilla, coherente, completa, concisa y, sobre todo, sin ambigüedades.

A continuación, se muestra un ejemplo de la manera en la que serán documentados los requisitos de software del sistema.

Tabla 4.43.: Ejemplo de documentación de un requisito de software.

RS_000	
Categoría	De documentación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	Los requisitos de software deben documentarse siguiendo este formato.

4.3.5. Identificación de requisitos de software

Tabla 4.44.: Requisito RS_001.

RS_001	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe permitir configurar los siguientes parámetros: 1. Dirección IP del computador en el que se encuentra instalado el sistema. 2. Directorio en el que se encuentran instalados los ficheros de configuración de Snort. 3. Rutas a posibles aplicaciones de apoyo de la aplicación. 4. Ruta en la que almacenar ficheros de apoyo de la aplicación.

Tabla 4.45.: Requisito RS_002.

RS_002	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe disponer de una sección dedicada a las operaciones de la aplicación Snort.

Tabla 4.46.: Requisito RS_003.

RS_003	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe disponer de una sección dedicada a las operaciones de la aplicación IPtables.

Tabla 4.47.: Requisito RS_004.

RS_004	
Categoría	De operación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe monitorizar la actividad de la aplicación Snort a través de la aplicación BASE.

Tabla 4.48.: Requisito RS_005.

RS_005	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar los parámetros de las siguientes variables de configuración de Snort: 1. Variables IP. 2. Variables de puertos.

Tabla 4.49.: Requisito RS_006.

RS_006	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar los parámetros de los siguientes preprocesadores de Snort: 1. Preprocesador Frag3. 2. Preprocesador Stream5. 3. Preprocesador sfPortscan. 4. Preprocesador HTTP Inspect. 5. Preprocesador Normalizer. 6. Preprocesador Reputation.

Tabla 4.50.: Requisito RS_007.

RS_007	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer junto a cada parámetro configurable de las variables una descripción emergente que informe acerca de qué valores pueden ser especificados.

Tabla 4.51.: Requisito RS_008.

RS_008	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer junto a cada parámetro configurable de los preprocesadores una descripción emergente que informe acerca de qué valores pueden ser especificados.

Tabla 4.52.: Requisito RS_009.

RS_009	
Categoría	De rendimiento.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema sólo debe mostrar las reglas que Snort aplica, sin mostrar aquellas reglas que, pese a estar especificadas en la aplicación, no se están aplicando.

Tabla 4.53.: Requisito RS_010.

RS_010	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe mostrar las reglas que Snort aplica organizadas según el fichero en el que se encuentren las mismas.

Tabla 4.54.: Requisito RS_011.

RS_011	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de lanzar la ejecución de la aplicación Snort.

Tabla 4.55.: Requisito RS_012.

RS_012	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de reiniciar la ejecución de la aplicación Snort.

Tabla 4.56.: Requisito RS_013.

RS_013	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de gestionar las siguientes tablas de la aplicación IPtables: 1. Tabla FILTER. 2. Tabla NAT. 3. Tabla MANGLE. 4. Tabla RAW.

Tabla 4.57.: Requisito RS_014.

RS_014	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de realizar las siguientes operaciones sobre las cadenas de las tablas de la aplicación IPtables: 1. Añadir una nueva cadena. 2. Renombrar una cadena.. 3. Eliminar una cadena. 4. Cambiar la política por defecto de una cadena.

Tabla 4.58.: Requisito RS_015.

RS_015	
Categoría	Funcional.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de realizar las siguientes operaciones sobre las reglas de las tablas de la aplicación IPtables: 1. Añadir una nueva regla. 2. Eliminar una regla..

Tabla 4.59.: Requisito RS_016.

RS_016	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de visualizar las reglas de cada tabla, organizadas según a la cadena a la que pertenezca cada una de ellas.

Tabla 4.60.: Requisito RS_017.

RS_017	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer la capacidad de visualizar las reglas de cada tabla en el orden en el que son aplicadas a los paquetes de red entrantes.

Tabla 4.61.: Requisito RS_018.

RS_018	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe ofrecer junto a cada parámetro configurable de la configuración de IPtables una descripción emergente que informe acerca de qué valores pueden ser especificados.

Tabla 4.62.: Requisito RS_019.

RS_019	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe indicar la fecha en la que se salvó por última vez el estado de la aplicación IPtables.

Tabla 4.63.: Requisito RS_020.

RS_020	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe indicar si se han realizado cambios en el estado de la aplicación IPtables que no han sido guardados.

Tabla 4.64.: Requisito RS_021.

RS_021	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe mostrar el estado de la aplicación que será ser restaurado.

Tabla 4.65.: Requisito RS_022.

RS_022	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe mostrar la fecha en la que se guardó el estado que puede ser restaurado.

Tabla 4.66.: Requisito RS_023.

RS_023	
Categoría	De comprobación.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema debe indicar de forma visual qué parámetros han tratado de ser actualizados con valores no permitidos.

Tabla 4.67.: Requisito RS_024.

RS_024	
Categoría	De interfaz.
Prioridad	Alta.
Estabilidad	Estable.
Verificabilidad	Alta.
Descripción	El sistema ofrecerá como método único de entrada formularios web.

4.4. Estudio de las alternativas de solución

4.4.1. Java Enterprise Edition (JEE)

Java Enterprise Edition es una plataforma de programación que permite desarrollar aplicaciones web multicapa independientes de la plataforma, es decir, la aplicación desarrollada con esta tecnología será independiente del servidor de aplicaciones en el que se ejecute la misma. Esta última característica es una de las más importantes que ofrece esta plataforma. Otra característica importante de la misma es que las aplicaciones desarrolladas son declarativas, es decir, existe una diferencia entre la forma de funcionar de las aplicaciones y el código de las mismas, existiendo mecanismos que permiten configurar el funcionamiento de ellas sin necesidad de modificar el código. Además, las aplicaciones desarrolladas harán uso de la infraestructura que proporciona el servidor en el que se ejecutan sin que el programador sea consciente de ello.

Por otra parte, esta plataforma ofrece un conjunto de interfaces de programación de aplicaciones o API's (del inglés Application Programming Interface), de manera que facilitan en gran medida el desarrollo de aplicaciones web. Estas API's ofrecen varias funcionalidades importantes. Podemos mencionar las que ofrecen funcionalidad para la conexión e interacción con una base de datos (Java Database Conectivity o JDBC), la comunicación de aplicaciones mediante colas de mensajes (Java Message Service o JMS) o la abstracción de orientación a objetos para construir webs dinámicas (Java Servlets), entre otras muchas.

Por último, la plataforma JEE permite desarrollar la aplicación de forma modular, lo que facilita la comprensión, el mantenimiento y la actualización de la misma, ya que el lenguaje Java sigue el paradigma de orientación a objetos (en el que se conciben los objetos como un conjunto de datos y métodos que manipulan dichos datos).

4.4.2. PHP Hypertext Pre-processor (PHP)

PHP es un lenguaje interpretado de alto nivel embebido en código HTML que nos permite modificar de forma dinámica el mismo. Se trata de un lenguaje ampliamente extendido en la red, aunque poco a poco está comenzando a ser desbancado [28], y existen multitud de librerías desarrolladas en este lenguaje que permiten implementar funcionalidades de manera muy simple. Esto quizá se debe a que puede ser utilizado sobre los principales sistemas operativos que se encuentran en el mercado (Windows, Mac OS, Linux, UNIX, etc), además de ser soportado por la mayoría de los servidores web disponibles (Apache, Microsoft Internet Information Server, etc).

Entre las características más destacables del lenguaje PHP es que éste puede seguir el paradigma de orientación a objetos, el paradigma procedural (en el que se especifican los pasos a seguir para resolver un problema específico) o llevar a cabo una mezcla de ambos.

Este lenguaje no tiene una sintaxis especialmente complicada, y permite el desarrollo de páginas web dinámicas que pueden hacer uso de conexiones a gran cantidad de bases de datos, lo cual supone otra de las características más importantes del lenguaje.

Además se trata de un lenguaje de código abierto (cuenta con licencia Open Source), lo que supone que los fallos en el funcionamiento del mismo sean detectados y reparados de forma rápida, además de que se encuentra en constante desarrollo. Esto también contribuye a que se creen nuevas librerías para este lenguaje que empleen nuevas funcionalidades de las que podemos hacer uso para el desarrollo de nuestra aplicación web. Entre las librerías disponibles podemos encontrar librerías para la creación de gráficos (como pChart o JpGraph), librerías para la integración con sistemas de pago (como PHP Payment Library) o librerías para la generación de ficheros PDF (Portable Document Format), entre muchas otras. Otra ventaja que presenta que el lenguaje cuente con esta licencia es que podemos adaptarlo a nuestras necesidades (habilitando únicamente las opciones que necesitemos).

4.4.3. Grails

Grails es un marco de trabajo (framework) que utiliza el lenguaje Groovy para el desarrollo ágil de aplicaciones web. Entre sus características principales podemos encontrar:

- Es un lenguaje de tipado dinámico, es decir, no es necesario especificar el tipo de dato de una variable a la hora de declararla (el entorno de ejecución es capaz de determinarlo en tiempo de ejecución).
- Permite acceder a todas las API's disponibles para Java.
- Puede ser empleado como lenguaje de scripting (sin necesidad de declarar clases).
- Puede ser ejecutado sobre la máquina virtual de Java (en inglés Java Virtual Machine, JVM) ya que cada clase declarada con este lenguaje se compila a código byte (más conocido por su término en inglés bytecode).

La principal ventaja de este framework es que Groovy es totalmente compatible con Java ("Cualquier cosa que pueda hacerse con Groovy es posible también con Java, y viceversa" [9]), por lo que puede ser utilizado para optimizar el proceso de desarrollo con JEE. Esta ventaja unida a las principales características del lenguaje Groovy mostradas anteriormente, y a las ventajas y características de JEE que expusimos en el primer apartado de esta sección, hacen de Grails una solución muy completa que debemos tener en cuenta.

A la hora de realizar el estudio de la tecnología a emplear, hay que tener presente que Groovy es un lenguaje de programación totalmente novedoso para nosotros, nunca hemos trabajado con él, aunque debido a que Groovy y Java son totalmente compatibles, cualquier dificultad puntual encontrada con Groovy puede ser solventada mediante Java. Por ejemplo, en el caso de no recordar, o no saber, cómo implementar cierta función con Groovy, podemos implementarla en Java sin ningún problema. Esto supone que la curva de aprendizaje, teniendo en cuenta que hemos trabajado con Java, es muy baja.

4.5. Valoración y elección de la solución

A la hora de elegir la tecnología a emplear para el desarrollo del proyecto, tenemos que tener en cuenta una serie de consideraciones, las cuales mostramos a continuación.

En primer lugar, consideramos la experiencia que tenemos en el manejo de cada una de las tecnologías presentadas. Con Java hemos trabajado en numerosas ocasiones, por lo que su uso es muy familiar para nosotros. Sin embargo, no ocurre lo mismo con PHP y Grails. Aunque también hemos trabajado en alguna ocasión con PHP, no podemos considerar que conozcamos este lenguaje más allá de su uso básico. Por otra parte, nunca habíamos trabajado con Grails, por lo que la experiencia con este marco de trabajo es nula. Pese a esto, debido a las expectativas que ofrecían las fuentes consultadas de que conociendo el lenguaje Java era bastante sencillo el uso del framework, lo investigamos haciendo pequeñas aplicaciones de prueba y ejemplos, ayudándonos en gran medida de [7], pero como ocurría en el caso de PHP no consideramos que nuestro conocimiento de esta tecnología sea más que el conocimiento básico de la misma. Con todo esto, la tecnología JEE toma una gran ventaja respecto a las demás.

Lo siguiente que consideramos es si la tecnología permite realizar una programación modular, de manera que la aplicación que desarrollemos pueda ser "compuesta" por módulos independientes, lo que facilitará en gran medida el desarrollo, el mantenimiento y la actualización de la misma. En este caso las tres tecnologías contempladas permiten una programación de este tipo.

Otra consideración tomada es si la tecnología es multiplataforma, es decir, si la aplicación desarrollada con dicha tecnología podrá funcionar sobre distintos sistemas operativos y arquitecturas. En este caso las tres tecnologías contempladas permiten desarrollar aplicaciones multiplataforma.

Por último, consideramos la posibilidad de usar un entorno de desarrollo con cada una de las tecnologías, puesto que este facilita el desarrollo de la aplicación poniendo a disposición del programador varias herramientas a través de una interfaz gráfica. En el caso de PHP y Grails, no disponemos de un entorno de desarrollo. Sin embargo para JEE disponemos de la herramienta Rational Software Architect for WebSphere, de la que se dará una pequeña descripción más adelante, con la que ya hemos trabajado en otras ocasiones.

Las consideraciones recogidas en esta sección se resumen en la Tabla (4.68).

Tabla 4.68.: Resumen de las consideraciones tenidas en cuenta a la hora de elegir la tecnología de desarrollo.

Tecnología	Experiencia	Prog. modular	Multiplataforma	Ent. de desarrollo
JEE	Sí	Sí	Sí	Sí
PHP	No	Sí	Sí	No
Grails	No	Sí	Sí	No

La conclusión que se obtiene de lo considerado en esta sección es que de las alternativas estudiadas para el desarrollo de la aplicación, la mejor es JEE.

4.5.1. Rational Software Architect for WebSphere

Rational Software Architect for WebSphere (o simplemente RSA) es un entorno de desarrollo, como ya mencionamos anteriormente, muy similar a Eclipse (de hecho está basada en este último) con el que también nos encontramos bastante familiarizados. El principal beneficio que aporta esta herramienta es que libera al programador de la aplicación de trabajos repetitivos como puede ser, por ejemplo, crear nuevos componentes de la aplicación que estamos desarrollando. De dichos componentes, nos proporcionará, una vez indicado el tipo y su nombre, su "esqueleto", permitiendo así al programador centrarse en el desarrollo de la funcionalidad de cada componente. En caso de no usar un entorno de desarrollo sería necesario crear todo el código del componente (y de todos los componentes) de forma manual.

El software RSA dispone de herramientas que permiten gestionar los proyectos web, navegar de forma rápida por los directorios del proyecto, detectar errores en la sintaxis del programa y ofrecer ayudas para solucionarlos, gestionar el despliegue de aplicaciones web sobre distintos servidores, realizar modelado de datos a través de UML, automatizar la creación de ciertas partes del código de la aplicación, visualizar la interfaz gráfica a medida que se va desarrollando, realizar depuración de código pudiendo establecer de forma rápida los puntos de parada en la ejecución, etc. Estos son sólo unos pocos ejemplos, ya que existe una gran cantidad de herramientas que pueden ser añadidas a este software. Se pueden consultar más características de RSA en [16].

Consideramos que este es un buen entorno de desarrollo con el que construir la aplicación web, tanto por todas las ventajas que aporta como por la experiencia que tenemos en su uso. Hay que tener en cuenta que esta herramienta no es gratuita, pero disponemos de una licencia académica para poder utilizarla, ya que dicha herramienta formó parte de una de las asignaturas cursadas anteriormente. Incluso si durante la fase de implementación de la aplicación web dicha licencia expirara, sería posible hacer uso del entorno de desarrollo Eclipse, en su versión Indigo, lo cual no sería ningún impedimento para el correcto desarrollo de la aplicación.

Capítulo 5.

Diseño del sistema

5.1. Definición de la arquitectura del sistema

El objetivo de esta sección es llevar a cabo un diseño de la arquitectura que el sistema a desarrollar debe cumplir.

Para el sistema a desarrollar se ha decidido que la arquitectura más apropiada es la conocida como Modelo-Vista-Controlador (más conocida por sus siglas en inglés MVC, Model-View-Controller). En esta arquitectura se distinguen las tres capas que se indican en el nombre de la misma que, de un modo general, se definen tal y como sigue [7]:

- La capa denominada Modelo contiene los componentes que representan y gestionan los datos manejados por la aplicación..
- La capa denominada Vista contiene los componentes responsables de mostrar al usuario el estado actual del modelo de datos, y presentar las distintas acciones disponibles.
- La capa denominada Controlador contiene los componentes que reciben las órdenes del usuario, gestionan la aplicación de la lógica de negocio sobre el modelo de datos, y determinan qué vista debe mostrarse a continuación.
    - Junto a esta última capa, existe una cuarta capa, normalmente denominada como Servicios, que es, en realidad, la que contiene los componentes encargados de implementar la lógica de negocio del sistema. La función de esta capa es disgregar las tareas realizadas por la lógica de negocio de las tareas realizadas por los componentes de la capa Controlador.

En la Figura (5.1) se muestra un esquema de los componentes, y la manera en la que interactúan los mismos, en el modelo MVC.

El flujo de control en los sistemas que disponen de esta arquitectura comienza en el momento en el que el usuario realiza una petición, la cual contiene los datos necesarios para desarrollar la acción que el usuario desea realizar, a un componente de la capa Vista. A continuación, dicho componente de la capa Vista genera una notificación que es enviada a un componente de la capa Controlador. Dentro de esta capa, el componente en cuestión delega en los componentes de la capa Servicios la lógica de negocio para que ésta modifique adecuadamente los componentes de la capa Modelo. Una vez que el

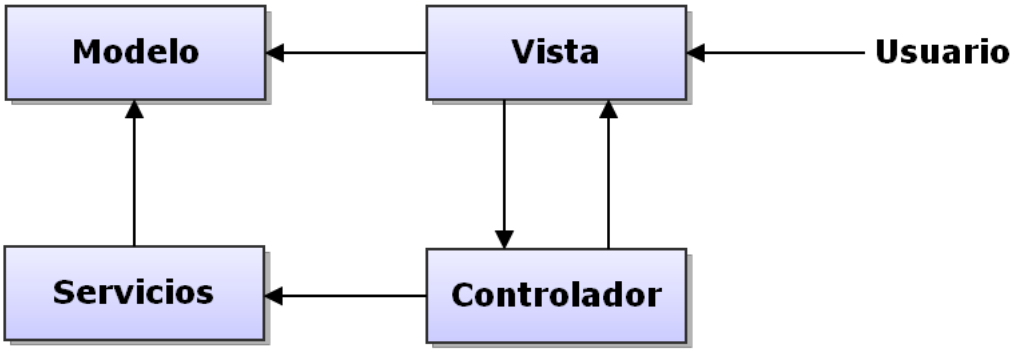

Figura 5.1.: Esquema de los componentes, y la manera en la que éstos interactúan, del modelo MVC.

componente de la capa Servicios a llevado a cabo su trabajo, el controlador vuelve a tomar el control de la ejecución y, tras haber comprobado el resultado de las operaciones realizadas por la capa Servicios, decide cuál es el componente de la capa Vista que debe ser mostrada al usuario. Por último, este componente genera, tomando los datos del modelo (sólo aquellos que resulten ser útiles en ese momento) la vista que será mostrada al usuario y es trasmitida al mismo. Este flujo de control se repite cuando el usuario vuelve a generar una nueva petición a un componente de la capa Vista.

La finalidad de esta arquitectura es lograr una agrupación de los componentes que conformarán el sistema final según el rol desempeñado por cada uno de ellos en la funcionalidad del mismo. Al realizar esta agrupación, se logra una independencia entre cada una de las agrupaciones de componentes surgidas. De esta forma, al agrupar los distintos componentes según su funcionalidad y que estas agrupaciones sean independientes las unas de las otras, resultará mucho más sencillo llevar a cabo tanto el mantenimiento, como las posibles actualizaciones del sistema. Esto es debido a que es posible realizar modificaciones en los componentes de una agrupación sin temer que dichas modificaciones afecten a los componentes presentes en el resto de agrupaciones.

Debido a las características presentadas en el párrafo anterior, el sistema resultante será sencillo de mantener y actualizar, sin embargo, resultará algo más complicado de desarrollar, ya que a lo largo de la fase de desarrollo será necesario, en primer lugar, tener en cuenta qué componentes se ubican en qué capas de la arquitectura, y en segundo lugar, se generará una cantidad considerable de ficheros fuente. Pese a esto, se considera que las ventajas que aporta la arquitectura compensan a las desventajas, resultando ser la arquitectura más apropiada para este sistema.

5.1.1. Identificación de subsistemas

Tras haber decidido la arquitectura que será implementada, y debido a que el sistema a desarrollar cumplirá la función de monitorizar y gestionar dos aplicaciones distintas (Snort e IPtables), en cada capa de la arquitectura será posible identificar módulos distintos en los que se incluirán, para lograr una mayor modularidad de la aplicación, los componentes necesarios para el manejo de sendas aplicaciones. En la Figura (5.2) se muestran cada uno de los módulos que contiene cada capa de la arquitectura.

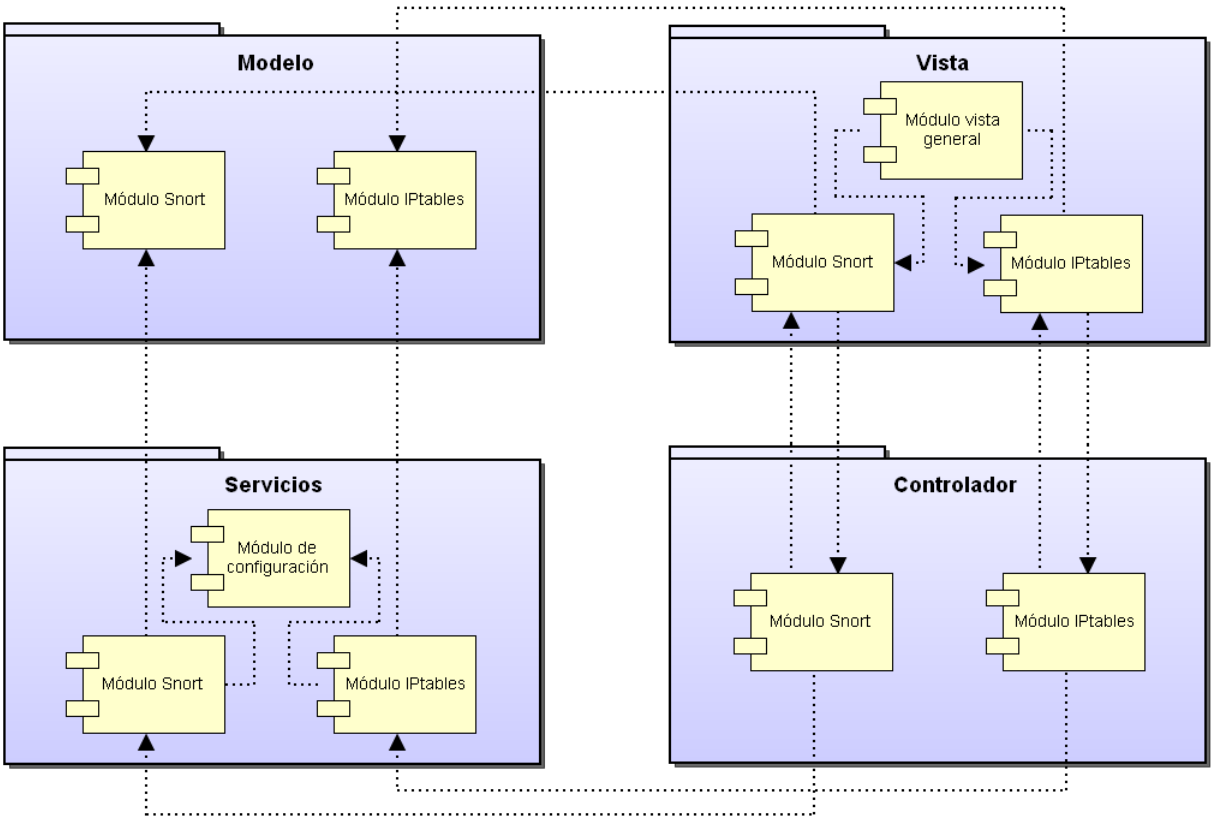

Figura 5.2.: Esquema de los módulos que conformarán el sistema.

En todas las capas de la arquitectura encontramos un módulo destinado a la gestión de Snort y otro módulo destinado a la gestión de IPtables. Parece lógico pensar que, al tratarse de dos aplicaciones totalmente distintas e independientes, resulta necesario separar su funcionalidad en módulos diferentes. De esta forma se logra que en el caso de producirse un fallo en alguno de los componentes (ya sea debido a errores de implementación o de actualización de los mismos) de un módulo, dicho fallo no afecte al correcto funcionamiento del otro módulo.

Además encontramos otros dos módulos en las capas de Vista y Servicios. El módulo que aparece en la capa Vista contendrá todas los componentes que no tengan relación con ninguna de las dos aplicaciones gestionadas, como podrían ser páginas de ayuda, de manuales o de configuración de la aplicación. El módulo que aparece en la capa Servicios

contendrá los componentes necesarios para acceder a los parámetros de configuración de la configuración, como podría ser la dirección IP del servidor. También contendrá los componentes necesarios para lanzar la ejecución de ciertos comandos en el entorno de la aplicación, como podría ser la ejecución de un comando para añadir una regla a una cadena de una tabla de IPtables.

5.1.2. Identificación de componentes

Llegados a este punto, en el que ya se han diseñado tanto las capas de la arquitectura, como los módulos que cada una de estas capas contendrán, hay que establecer qué tipo de componentes será necesario utilizar en cada uno de estos módulos.

En primer lugar, en cada uno de los módulos de la capa Modelo, puesto que trabajaremos con un lenguaje de programación orientado a objetos, será necesario emplear objetos, los cuáles serán diseñados para lograr mantener el estado de cada una de las aplicaciones que el sistema gestionará. Debemos destacar en este ámbito que para implementar la funcionalidad requerida de la aplicación usaremos objetos Java sencillos, es decir, no haremos uso de objetos más avanzados como, por ejemplo, EJBs. El principal motivo de esta decisión es que los EJBs precisan ser ejecutados en un contenedor de EJBs. Dicho contenedor sólo está presente en servidores que requieren mayores cantidades de memoria RAM para ejecutar, mayor tiempo para ejecutar el servidor, etc. Con la finalidad de que la aplicación pueda ser instalada requiriendo simplemente un servidor que disponga de un contenedor web (estos servidores son mucho más "ligeros" en cuanto a requisitos de RAM y tiempo de arranque), y puesto que en este caso consideramos que las ventajas de utilizar EJBs no son significativas, optamos por utilizar, como ya hemos explicado, objetos Java sencillos. Más adelante mostraremos el diseño de los objetos que serán necesarios para mantener el estado de Snort e IPtables.

En cada uno de los módulos de la capa Vista las tecnologías que emplearemos serán HTML para mostrar los contenidos estáticos de la aplicación (aquellos contenidos que no cambian independientemente de las acciones realizadas por el usuario de la aplicación) y, puesto que trabajaremos con la plataforma de desarrollo Java, JSP para mostrar los contenidos dinámicos de la aplicación (aquellos contenidos que varían en función de las acciones realizadas por el usuario de la aplicación). Para definir el estilo de la interfaz de la aplicación utilizaremos Cascade Style Sheet (CSS), ya que esta tecnología permitirá abstraer las cuestiones relacionadas con la visualización de la interfaz (colores, tamaños de letra, división de la pantalla, etc.).

En cada uno de los módulos de la capa Controlador, será utilizada la tecnología de Java Servlets. Dicha tecnología proporciona componentes, también escritos en Java, que permiten extender la capacidad de respuesta a las peticiones realizadas por los clientes. Estos componentes se adaptan muy bien a la funcionalidad que se supone que la capa Controlador debe ofrecer, puesto que a través de ellos el sistema recibe la petición generada por el usuario de la aplicación, y además, añadiendo unas sencillas líneas de código, se logra la funcionalidad de elegir la operación que se ha de llevar a cabo, y tras haber sido realizada ésta seleccionar el componente de la capa Vista que se debe mostrar al usuario.

Por último, en cada uno de los módulos de la capa Servicios, será necesario emplear, al igual que en la capa Modelo, objetos Java sencillos. Sin embargo, y puesto que estos objetos no tienen como función almacenar ningún estado, las clases que los implementen sólo contarán con los métodos estáticos necesarios para modificar los objetos que sí mantienen el estado de Snort e IPtables.

En la Figura (5.3) se muestra un resumen de cada uno de los componentes de cada capa de la arquitectura.

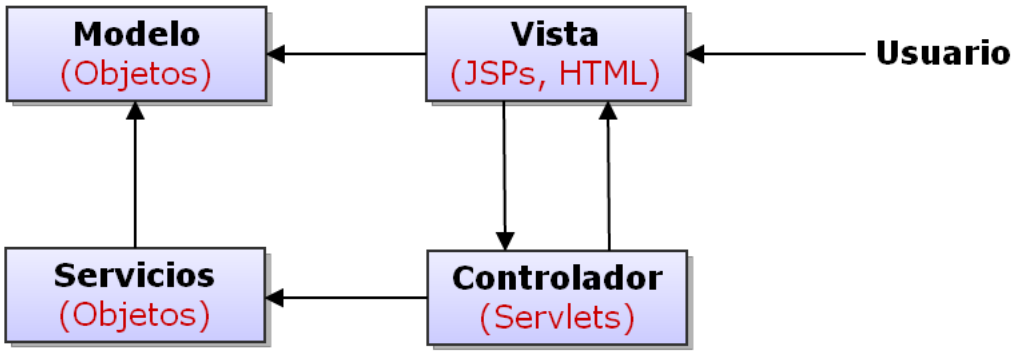

Figura 5.3.: Resumen del tipo de componentes para cada capa de la arquitectura MVC.

5.2. Diseño de clases

El objetivo de esta sección es identificar las clases que componen el modelo de clases de diseño, partiendo del diseño de la arquitectura del sistema realizado en secciones anteriores de este mismo capítulo. Las clases diseñadas se irán agrupando en los siguientes apartados de esta sección en función de la capa y del módulo al que deben pertenecer. En el caso de la capa Vista, puesto que se empleará, tal y como se expuso anteriormente, la tecnología JSPs y HTML, no encontraremos clases ni objetos con los que trabajar, por lo que no resulta necesario incluir el diseño de dicha capa en esta sección.

5.2.1. Capa Modelo

Módulo IPtables

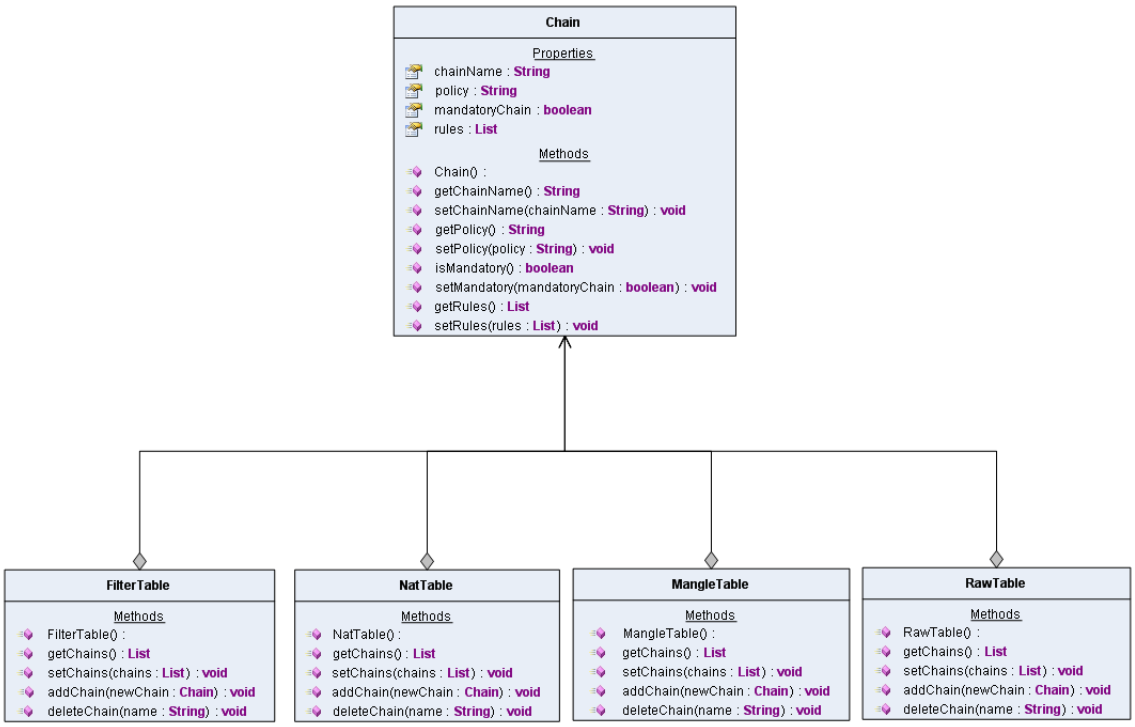

Figura 5.4.: Diseño de clases para el módulo IPtables de la capa Modelo.

Módulo Snort

Figura 5.5.: Diseño de clases para el módulo Snort de la capa Modelo (Parte I).

Figura 5.6.: Diseño de clases para el módulo Snort de la capa Modelo (Parte II).

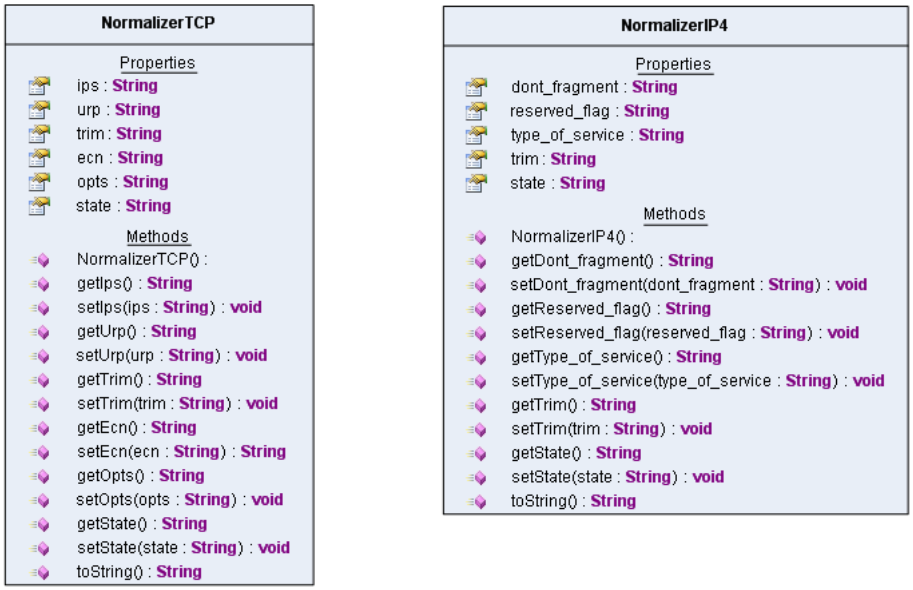

Figura 5.7.: Diseño de clases para el módulo Snort de la capa Modelo (Parte III).

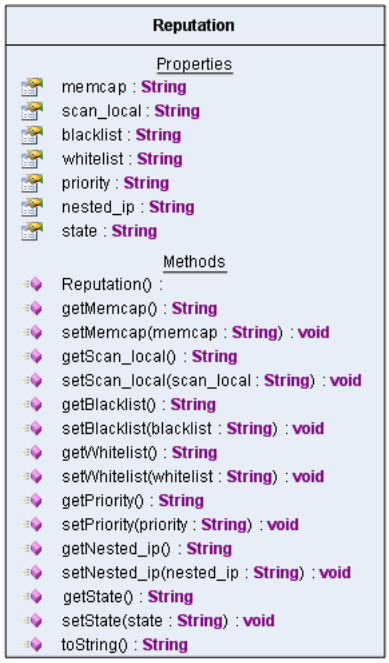

Figura 5.8.: Diseño de clases para el módulo Snort de la capa Modelo (Parte IV).

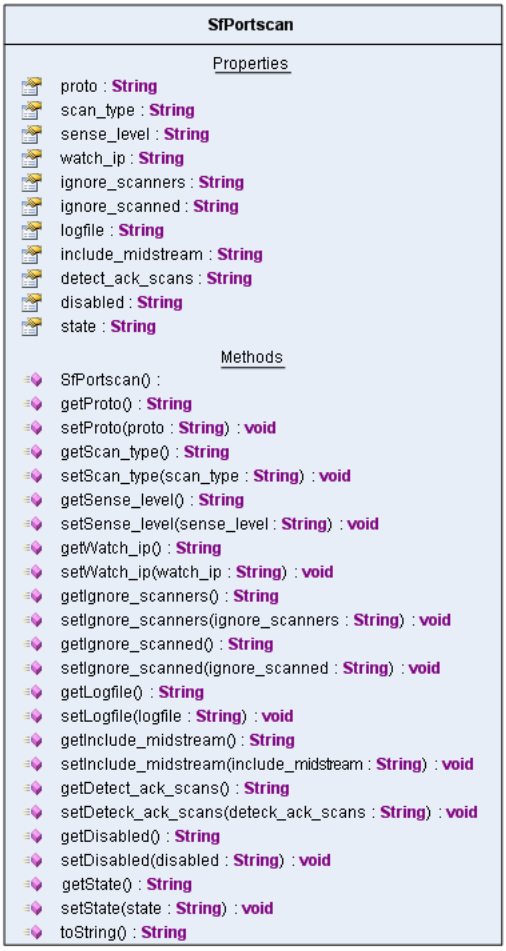

Figura 5.9.: Diseño de clases para el módulo Snort de la capa Modelo (Parte V).

Figura 5.10.: Diseño de clases para el módulo Snort de la capa Modelo (Parte VI).

Figura 5.11.: Diseño de clases para el módulo Snort de la capa Modelo (Parte VII).

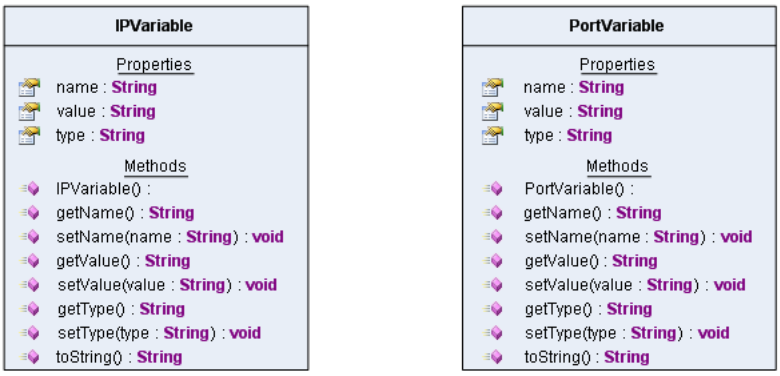

Figura 5.12.: Diseño de clases para el módulo Snort de la capa Modelo (Parte VIII).

5.2.2. Capa Controlador

Módulo IPtables

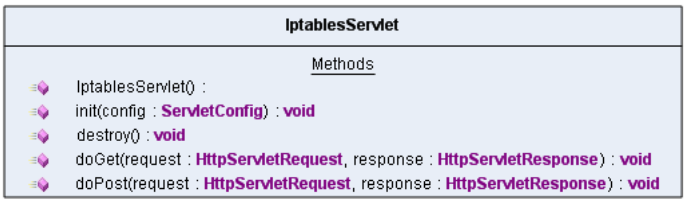

Figura 5.13.: Diseño de clases para el módulo IPtables de la capa Controlador.

Módulo Snort

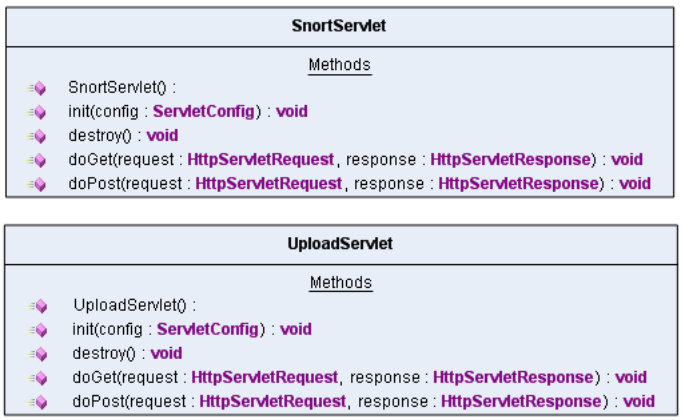

Figura 5.14.: Diseño de clases para el módulo Snort de la capa Controlador.

5.2.3. Capa Servicios

Módulo de configuración

Figura 5.15.: Diseño de clases para el módulo de configuración de la capa Servicios.

Módulo IPtables

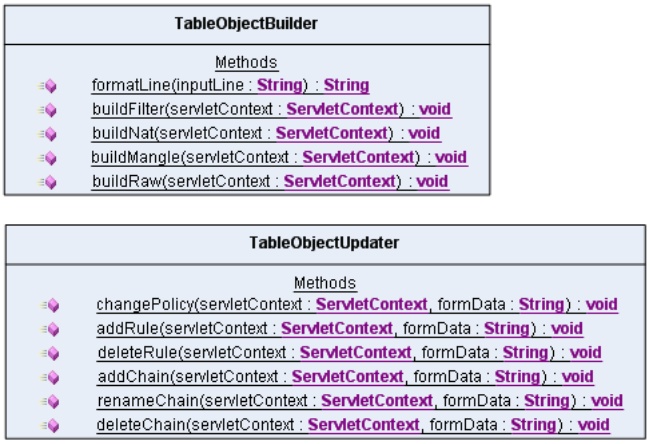

Figura 5.16.: Diseño de clases para el módulo IPtables de la capa Servicios.

Módulo Snort

Figura 5.17.: Diseño de clases para el módulo Snort de la capa Servicios (Parte I).

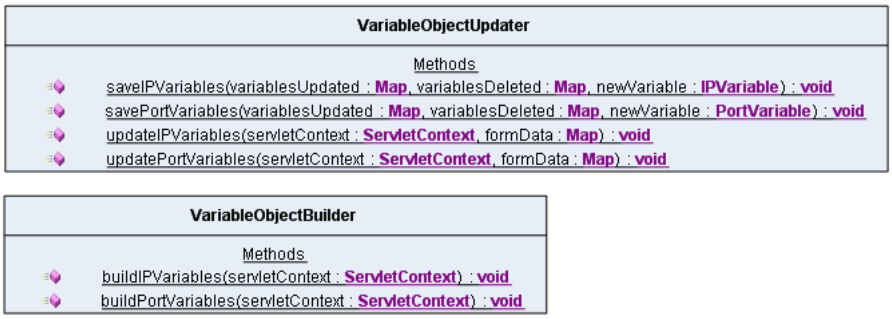

Figura 5.18.: Diseño de clases para el módulo Snort de la capa Servicios (Parte II).

5.3. Diseño de la interfaz gráfica

La finalidad de esta sección es realizar el diseño de la interfaz con la que contará el sistema que se está desarrollando y a través de la cual el usuario podrá llevar a cabo su labor. A través de este diseño quedará establecida tanto la apariencia con la que contarán cada una de las pantallas de la aplicación a través de las que el usuario puede navegar, como la forma en la que trabaja dicha interfaz.

5.3.1. Identificación de las tareas a realizar

Tras haber identificado las capacidades técnicas de los futuros usuarios de la aplicación en la sección 4.1, nos centramos en qué tareas debe llevar a cabo la misma.

Como se ha mencionado en anteriores ocasiones, el sistema deberá monitorizar y gestionar la actividad de las aplicaciones IPtables y Snort. Tanto la monitorización como la gestión de ambas aplicaciones suele realizarse a través de comandos de consola y/o de ficheros de configuración con una gran cantidad de líneas que permiten especificar el funcionamiento de las mismas, lo que puede hacer del trabajo del administrador de estas aplicaciones algo tedioso y complicado.

Las principales tareas que el sistema ofrecerá para la monitorización y gestión de IPtables serán:

- Establecer la acción a aplicar para los paquetes que no cumplan ninguna de las reglas especificadas en las cadenas de todas las tablas ofrecidas por la aplicación.
- Añadir reglas a las cadenas de todas las tablas ofrecidas por la aplicación.
- Eliminar reglas de las cadenas de todas las tablas ofrecidas por la aplicación.
- Listar las reglas contenidas en las cadenas de todas las tablas ofrecidas por la aplicación.
- Añadir cadenas a todas las tablas ofrecidas por la aplicación.
- Renombrar cadenas de todas las tablas ofrecidas por la aplicación (sólo de aquellas que han sido añadidas por el usuario).
- Eliminar cadenas de todas las tablas ofrecidas por la aplicación (sólo de aquellas que han sido añadidas por el usuario).

Por otra parte, las principales tareas que el sistema ofrecerá para la monitorización y gestión de Snort serán:

- Añadir variables[1].
- Modificar variables.

[1] Tanto las denominadas IP variables como las denominadas Port variables

- Eliminar variables (sólo de aquellas que han sido añadidas por el usuario).
- Modificar los parámetros de los preprocesadores frag3, stream5, sfPortscan, HTTPInspect, Normalizer y Reputation.
- Visualizar información de apoyo para una correcta configuración de cada uno de los parámetros de los preprocesadores.
- Visualizar las reglas de detección establecidas en la aplicación.
- Visualizar las alertas generadas por la aplicación.

Para lograr un sistema usable y útil para el usuario, el posterior diseño de la interfaz gráfica deberá realizarse teniendo en mente tanto las tareas identificadas en esta sección, como el perfil de usuario especificado en el capítulo anterior.

5.3.2. Diseño de la interfaz

Puesto que a través de la interfaz que se diseñará será la forma de que el usuario final de la aplicación interactúe con el sistema, dicha interfaz tendrá gran parte de responsabilidad (o culpa) de que el usuario perciba, o no, la utilidad del sistema. Por tanto, resulta muy importante lograr un diseño adecuado para la interfaz del mismo.

En [19] el autor propone tres reglas de oro para el diseño de una buena interfaz, así como una serie de principios para cada una de estas reglas. A continuación se muestran dichas reglas, así como una breve descripción del significado de cada una de ellas[2]:

1. Dar el control al usuario. El significado de esta regla es que el usuario debe percibir que la aplicación cubre sus necesidades y que le ayuda a desarrollar su trabajo. Dicho de otra forma, siente que domina la aplicación y no al contrario.

2. Reducir la carga de memoria del usuario. El significado de esta regla es que se debe evitar, en la medida de lo posible, que el usuario se vea obligado a recordar demasiadas cosas relacionadas con el funcionamiento de la aplicación puesto que, cuantas más tenga que recordar, mayor probabilidad existe de que cometa errores. Dicho de otro modo, una interfaz de usuario bien diseñada no depende de la memoria del mismo.

3. Lograr que la interfaz sea consistente. El significado de esta regla es que toda la información mostrada por la interfaz debe: mostrar la misma organización, los mecanismos de entrada de datos, así como los de navegación, deben ser lo más restringidos posibles y mantenerse a lo largo de todas las partes de la aplicación.

Además de tener en cuenta estas directrices a la hora de realizar el diseño, tomaremos como ayuda otras aplicaciones web, y su diseño de interfaz gráfica, que ya se encuentran

[2] En la enumeración de cada una de las reglas se han omitido cada uno de los principios de diseño que ayudan a cumplirlas puesto que ésta no es la finalidad de este documento.

desarrolladas y con las que los futuros usuarios de la aplicación puede que hayan trabajado o que se encuentren familiarizados. Estas aplicaciones a las que hacemos referencia son, en primer lugar, algunas consolas de administración de servidores web o servidores de aplicaciones (como las que se muestran en la Figura (5.19)) o aplicaciones tan relacionadas con el tema que nos ocupa como BASE (tal y como se muestra en la Figura (5.20)).

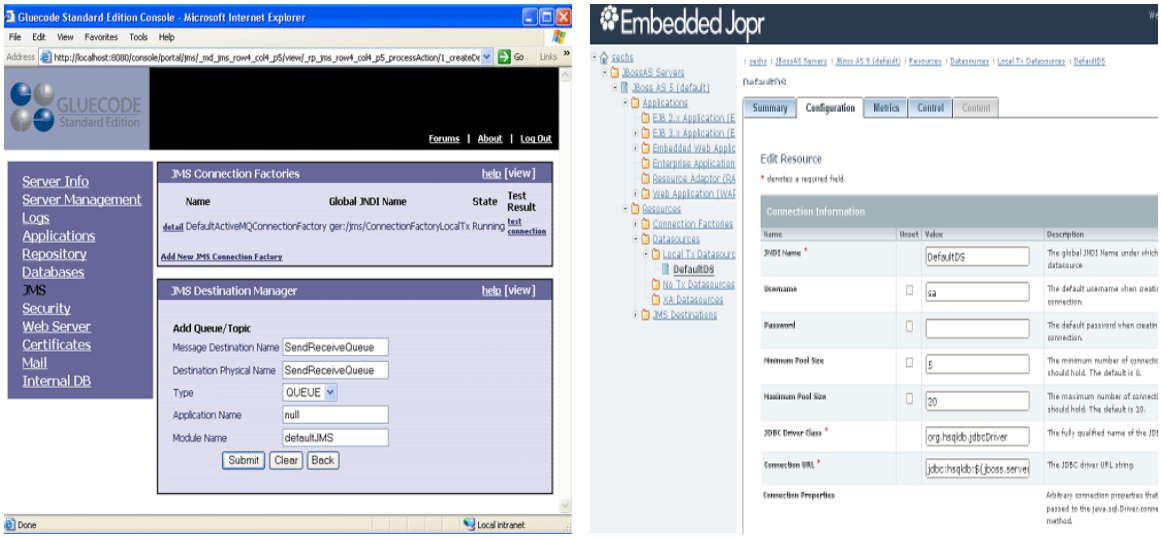

Figura 5.19.: Interfaz gráfica de consolas de administración de servidores web.

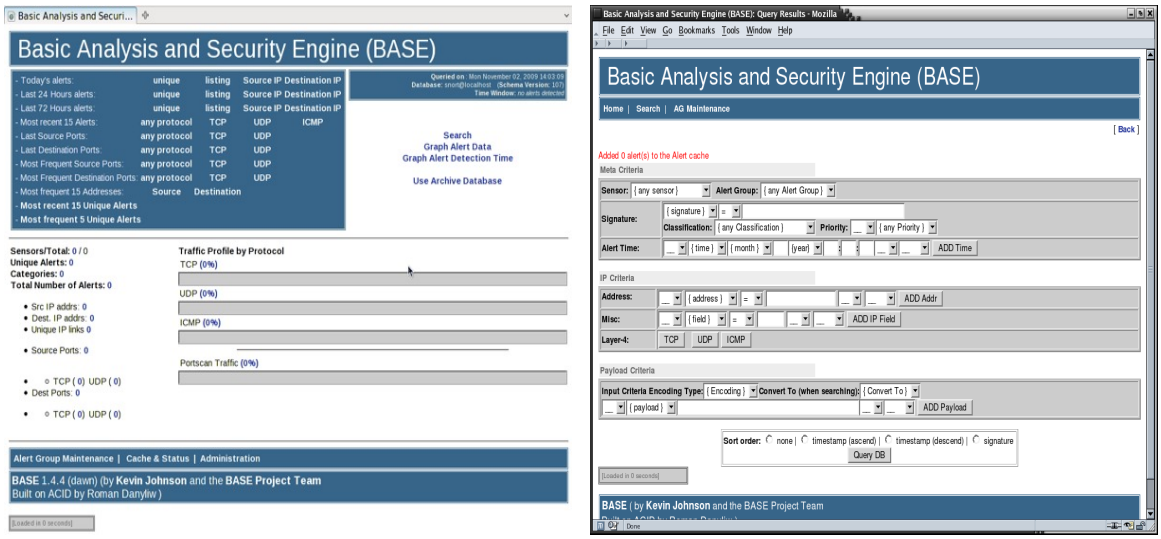

Figura 5.20.: Interfaz gráfica de BASE.

Tras estas consideraciones, el diseño de la interfaz gráfica deberá cumplir las siguientes condiciones:

- El estilo de la interfaz será sencillo y directo. Con esta condición se asegura que la interfaz no cuente con demasiadas imágenes o elementos que ralenticen la carga de la misma (ya que el sistema servirá para que el usuario lleve a cabo su trabajo, resulta más conveniente ofrecerle la posibilidad de hacerlo de un modo rápido).

- Todas las pantallas que conformen la interfaz de la aplicación se ajustarán al mismo formato. Con esta condición se asegura que la carga de memoria del usuario es mínima, puesto que al mantener el mismo patrón en la interfaz, el usuario recordará para que sirve cada sección de la misma, y a qué lugar debe dirigirse para realizar la acción que desea.

- En todo momento se ofrecerá información al usuario acerca de en qué parte de la aplicación se encuentra. Con esta condición se asegura que el usuario tiene siempre presente el lugar en el que se encuentra y cómo puede navegar hacia cualquier otro lugar.

- En todo momento se ofrecerá al usuario, independientemente de en qué parte de la aplicación se encuentre, acceso a una parte que contenga información de apoyo para el uso de la aplicación. Con esta condición se asegura que el usuario, en caso de necesitarlo, puede obtener información de ayuda para llevar a cabo su labor.

- La forma en la que se recogerán los datos de entrada del usuario serán, en cualquier caso, formularios. Con esta condición se asegura que el usuario no debe memorizar la forma en la que se le pedirá que introduzcan datos en las distintas partes de la aplicación, es decir, esta condición también reduce la carga de memoria del usuario.

- La interfaz ofrecerá ayuda al usuario sobre qué valores puede introducir en cada campo de los formularios. Con esta condición se asegura que el usuario queda liberado de la tarea de acceder al manual de la aplicación con la que está trabajando y buscar cuáles son los valores más convenientes para cada campo del formulario.

- La interfaz indicará, en caso de error del usuario al introducir valores en el formulario, cuáles son los campos que contienen valores incorrectos. Con esta condición se asegura que el usuario puede identificar de forma rápida dónde ha cometido el error.

- En todo momento se ofrecerá información al usuario acerca de qué versión del sistema está utilizando, en que fecha ha sido publicado e información de contacto con el desarrollador. Con esta condición se asegura que el usuario está informado de si utiliza la versión más actualizada del sistema y tiene una vía de contacto para reportar cualquier sugerencia y/o error en el funcionamiento del sistema.

Por otra parte, una las consideraciones más importantes en esta parte del diseño de la aplicación, recae en la gestión de los posibles errores y/o excepciones que pueden aparecer

a lo largo del funcionamiento del sistema. Puesto que la información que ofrecen por defecto la mayoría de los servidores web al producirse un error/excepción no es demasiado significativa (se suelen limitar a indicar la excepción que ha ocurrido y los puntos en los que apareció la excepción), resultará de gran ayuda ampliar dicha información. Para este fin, será necesario añadir un diseño para las páginas de error. La siguiente información puede resultar de gran ayuda a la hora de identificar el motivo del fallo de la aplicación:

- Código de error. Este código permitirá identificar de manera única el motivo por el cual la ejecución de la aplicación produjo el fallo.
- Descripción. Consistirá en una breve descripción del fallo que se produjo.
- Posibles motivos. Se mostrará una lista con los posibles motivos por los cuales se pudo haber producido el fallo. Ofreciendo esta información, al usuario final o al administrador de la web, le resultará mucho más sencillo identificar el motivo por el que se ha producido el fallo y actuar en consecuencia.

Finalmente, se muestra un bosquejo de la estructura general de la interfaz diseñada siguiendo todas las consideraciones planteadas tanto en esta sección, como en secciones anteriores. También se ofrece la estructura de algunas de las principales pantallas que conformarán la interfaz de la aplicación web. Se ha evitado mostrar todas las pantallas que conformarán la interfaz, puesto que todas ellas siguen la estructura general mostrada a continuación, por lo que simplemente se ofrece en la Figura (5.22), a modo de ejemplo, dos casos concretos de aplicación de la estructura general (Figura (5.21)).

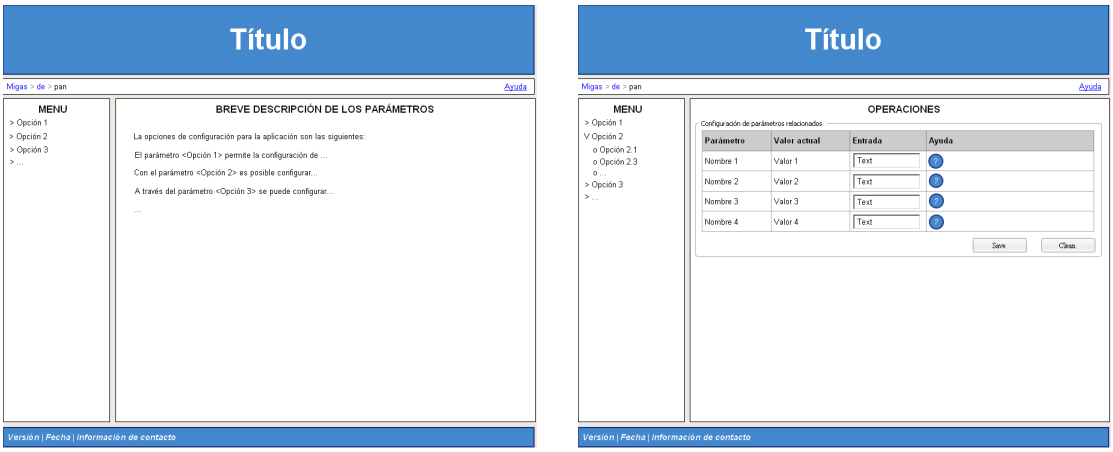

Figura 5.21.: Estructura general para las pantallas de la interfaz de la aplicación.

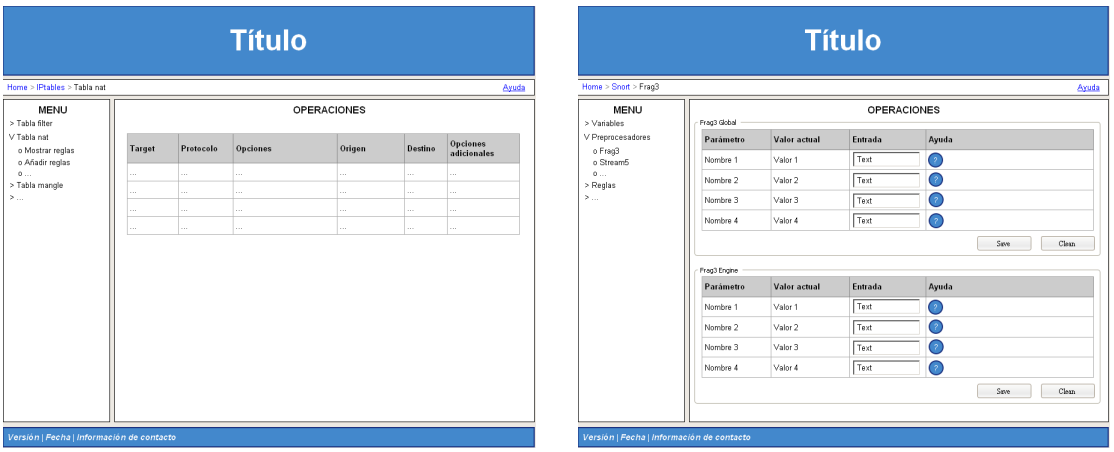

Figura 5.22.: Estructura concreta para dos de las pantallas de la interfaz de la aplicación.

Capítulo 6.

Pruebas y evaluación

6.1. Pruebas del sistema

En esta sección se elaborará un plan de pruebas que definirá una serie de acciones que serán llevadas a cabo durante el desarrollo del sistema con el objetivo de comprobar, en todo momento, si el mismo cumple con las necesidades expresadas por el cliente. Esto significa que el plan de pruebas que posteriormente se desarrollará servirá para comprobar que el sistema hace estrictamente lo especificado en los requisitos de usuario.

Además de especificar las pruebas que se han de realizar, también se proporcionarán los resultados que el sistema debe ofrecer, para que posteriormente sea posible verificar si el sistema cumple con los requisitos para pasar dicha prueba, y para facilitar el trabajo de comprobar si el resultado que ofrece el sistema coincide con lo que se esperaba.

6.1.1. Especificación del plan de pruebas

Sobre el sistema se llevarán a cabo pruebas a distintos niveles. Los tres niveles de pruebas a aplicar serán:

- Pruebas unitarias. Este nivel de pruebas se aplica sobre cada componente del sistema. El objetivo principal de las pruebas unitarias es verificar la funcionalidad y estructura de cada componente individual del sistema.
- Pruebas de integración. Este nivel de pruebas se aplica sobre cada conjunto de componentes, es decir, sobre cada uno de los módulos que componen el sistema y que fueron especificados en la sección 5.1. El objetivo de las pruebas de integración es verificar el correcta conexión entre los distintos componentes.
- Pruebas de sistema. Este nivel de pruebas se aplica sobre el sistema completo y sobre los sistemas que están relacionados con el mismo. El objetivo de las pruebas del sistema es verificar el correcto funcionamiento del sistema en su totalidad, y su correcta operación con los sistemas con los que interactúa.

6.1.2. Pruebas unitarias

La documentación de las pruebas incluidas dentro del nivel de pruebas unitarias se realizará a través de los siguientes atributos:

- Identificador. Cada prueba dispondrá de un identificador único que la represente de manera unívoca. Este atributo será PRU_XXX, resultando necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar la prueba unitaria.
- Objetivo. Este atributo describirá el fin de la prueba a realizar.
- Capa. Este atributo identificará sobre qué capa de la arquitectura se está aplicando la prueba en cuestión. Este atributo podrá tomar los siguientes valores:
  - Modelo.
  - Vista.
  - Controlador.
  - Servicios.
- Clase. Este atributo identificará sobre qué clase del modelo de clases se está aplicando la prueba en cuestión. Este atributo podrá tomar como valor cualquiera de los nombres de las clases obtenidas durante el diseño del sistema.
- Métodos. Este atributo identificará los nombres de los métodos de la clase sobre los que se está aplicando la prueba en cuestión. Este atributo podrá tomar como valor cualquiera de los nombres de los métodos obtenidos durante el diseño del sistema.
- Precondiciones. Este atributo identificará las condiciones que se deben cumplir antes de realizar la prueba.
- Entrada. Este atributo identificará el valor que se proporcionará a la prueba para comprobar el comportamiento del componente.
- Salida. Este atributo identificará el valor, o conjunto de valores, válido que el componente debe devolver para comprobar el comportamiento del mismo.

A continuación, se muestra un ejemplo de la manera en la que serán documentados las pruebas unitarias.

Tabla 6.1.: Ejemplo de documentación de una prueba unitaria.

PRU_000	
Objetivo	Mostrar un ejemplo de cómo documentar las pruebas unitarias.
Capa	Modelo.
Clase	Frag3Engine.
Métodos	Frag3Engine.
Precondiciones	-
Entrada	Las pruebas unitarias deben documentarse siguiendo este formato.
Salida	Todas las pruebas unitarias deben seguir este formato.
Secuencia	Redactar los requisitos de usuario en este formato.

Identificación de pruebas unitarias

Tabla 6.2.: Prueba unitaria PRU_001.

PRU_001	
Objetivo	Comprobar que el sistema lanza correctamente la aplicación Snort.
Capa	Servicios.
Clase	Launcher.
Métodos	launchSnort.
Precondiciones	La aplicación Snort no se encuentra en estado de ejecución.
Entrada	-
Salida	La aplicación Snort se encuentra en estado de ejecución.

Tabla 6.3.: Prueba unitaria PRU_002.

PRU_002	
Objetivo	Comprobar que el sistema reinicia correctamente la aplicación Snort.
Capa	Servicios.
Clase	Launcher.
Métodos	launchSnort.
Precondiciones	La aplicación Snort se encuentra en estado de ejecución.
Entrada	-
Salida	La ejecución de la aplicación Snort es detenida y después se vuelve a ejecutar.

Tabla 6.4.: Prueba unitaria PRU_003.

PRU_003	
Objetivo	Comprobar que el sistema accede correctamente al estado de la aplicación Snort.
Capa	Servicios.
Clase	Launcher.
Métodos	checkSnortStatus.
Precondiciones	La aplicación de Snort se encuentra parada.
Entrada	-
Salida	El sistema indica que la aplicación Snort se encuentra parada.

Tabla 6.5.: Prueba unitaria PRU_004.

PRU_004	
Objetivo	Comprobar que el sistema accede correctamente al estado de la aplicación Snort.
Capa	Servicios.
Clase	Launcher.
Métodos	checkSnortStatus.
Precondiciones	La aplicación de Snort se encuentra en ejecución.
Entrada	-
Salida	El sistema indica que la aplicación Snort se encuentra en ejecución.

Tabla 6.6.: Prueba unitaria PRU_005.

PRU_005	
Objetivo	Comprobar que el sistema modifica correctamente la política por defecto de las cadena de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Métodos	iptablesChangePolicy.
Precondiciones	La aplicación de IPtables cuenta con la cadena en la tabla de la que se va a modificar la política.
Entrada	Nombre de la tabla, nombre de la cadena y política que se desea establecer.
Salida	La cadena de la tabla especificada cambia su política por defecto a la especificada en la entrada.

Tabla 6.7.: Prueba unitaria PRU_006.

PRU_006	
Objetivo	Comprobar que el sistema accede correctamente a las reglas de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Métodos	iptablesSeeRules.
Precondiciones	La aplicación de IPtables cuenta, en la tabla a la que se va a acceder, con una serie de reglas.
Entrada	Nombre de la tabla.
Salida	El sistema muestra todas las reglas en el mismo orden que las muestra el comando iptables -t [TABLA] -L

Tabla 6.8.: Prueba unitaria PRU_007.

PRU_007	
Objetivo	Comprobar que el sistema añade correctamente reglas a las cadenas de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Métodos	iptablesAddRule.
Precondiciones	La aplicación de IPtables cuenta, en la tabla a la que se va a acceder, con la cadena a la que se le va a añadir la regla.
Entrada	Nombre de la tabla y regla que se desea añadir.
Salida	La cadena de la tabla a la que se ha añadido la regla cuenta con dicha regla.

Tabla 6.9.: Prueba unitaria PRU_008.

PRU_008	
Objetivo	Comprobar que el sistema elimina correctamente reglas de las cadenas de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Métodos	iptablesDeleteRule.
Precondiciones	La aplicación de IPtables cuenta, en la tabla a la que se va a acceder, con la cadena de la que se va a eliminar la regla y con la regla que va a ser eliminada.
Entrada	Nombre de la tabla, nombre de la cadena e índice de la regla que va a ser eliminada.
Salida	La cadena de la tabla de la que se ha eliminado la regla ya no cuenta con dicha regla.

Tabla 6.10.: Prueba unitaria PRU_009.

PRU_009	
Objetivo	Comprobar que el sistema elimina correctamente todas reglas de las cadenas de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Métodos	iptablesDeleteAllRules.
Precondiciones	La aplicación de IPtables cuenta, en la tabla a la que se va a acceder, con la cadena de la que se va a eliminar la regla y con algunas reglas.
Entrada	Nombre de la tabla y nombre de la cadena.
Salida	La cadena de la tabla de la que se ha eliminado no contiene ninguna regla.

Tabla 6.11.: Prueba unitaria PRU_010.

PRU_010	
Objetivo	Comprobar que el sistema añade correctamente cadenas a las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Métodos	iptablesAddChain.
Precondiciones	La aplicación de IPtables cuenta con la tabla a la que se va a acceder.
Entrada	Nombre de la tabla y nombre de la cadena.
Salida	La tabla cuenta con una nueva cadena cuyo nombre es el que se ha especificado en la entrada.

Tabla 6.12.: Prueba unitaria PRU_011.

PRU_011	
Objetivo	Comprobar que el sistema renombra correctamente cadenas a las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Método	iptablesRenameChain.
Precondiciones	La aplicación de IPtables cuenta con la tabla y la cadena a las que se va a acceder.
Entrada	Nombre de la tabla, antiguo nombre de la cadena y nuevo nombre de la cadena.
Salida	La tabla cuenta con una cadena cuyo nombre es el nuevo nombre de la cadena que se ha especificado en la entrada, y no cuenta con ninguna cadena cuyo nombre es el antiguo nombre especificado en la entrada.

Tabla 6.13.: Prueba unitaria PRU_012.

PRU_012	
Objetivo	Comprobar que el sistema elimina correctamente cadenas de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Método	iptablesDeleteChain.
Precondiciones	La aplicación de IPtables cuenta con la tabla a la que se va a acceder y con la cadena que se va a eliminar.
Entrada	Nombre de la tabla y nombre de la cadena.
Salida	La tabla no cuenta con ninguna cadena cuyo nombre sea el que se ha especificado en la entrada.

Tabla 6.14.: Prueba unitaria PRU_013.

PRU_013	
Objetivo	Comprobar que el sistema guarda correctamente el estado de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Método	saveIptablesState.
Precondiciones	-
Entrada	Contexto de servlet de la aplicación y ruta del fichero en el que almacenar el estado.
Salida	El estado de la aplicación IPtables queda almacenado en el fichero ubicado en la ruta especificada en la entrada.

Tabla 6.15.: Prueba unitaria PRU_014.

PRU_014	
Objetivo	Comprobar que el sistema restaura correctamente el estado de la aplicación IPtables.
Capa	Servicios.
Clase	Launcher.
Método	restoreIptablesState.
Precondiciones	Se ha guardado el estado de la aplicación IPtables.
Entrada	Contexto de servlet de la aplicación y ruta del fichero en el que se encuentra almacenado el estado.
Salida	El estado de la aplicación IPtables coincide con el que se encuentra almacenado en el fichero ubicado en la ruta especificada en la entrada.

Tabla 6.16.: Prueba unitaria PRU_015.

PRU_015	
Objetivo	Comprobar que el sistema es capaz de recuperar la salida ofrecida por la ejecución de comandos de consola.
Capa	Servicios.
Clase	ReaderStream.
Método	run.
Precondiciones	-
Entrada	-
Salida	La instancia de la clase ReaderStream almacena en su atributo output la salida que ofrece el comando que ha sido ejecutado por el sistema.

Tabla 6.17.: Prueba unitaria PRU_016.

PRU_016	
Objetivo	Comprobar que el sistema es capaz de recuperar parámetros de configuración del sistema.
Capa	Servicios.
Clase	ManagerConfiguration.
Método	getParameter.
Precondiciones	Al menos uno de los parámetros de configuración del sistema ha sido establecido.
Entrada	Nombre del parámetro.
Salida	El método devuelve el valor que fue especificado en la configuración del sistema.

Tabla 6.18.: Prueba unitaria PRU_017.

PRU_017	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado de la tabla filter de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectBuilder.
Método	buildFilter.
Precondiciones	-
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye el objeto Filter con los parámetros establecidos en la configuración de IPtables.

Tabla 6.19.: Prueba unitaria PRU_018.

PRU_018	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado de la tabla nat de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectBuilder.
Método	buildNat.
Precondiciones	-
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye el objeto Nat con los parámetros establecidos en la configuración de IPtables.

Tabla 6.20.: Prueba unitaria PRU_019.

PRU_019	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado de la tabla mangle de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectBuilder.
Método	buildMangle.
Precondiciones	-
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye el objeto Mangle con los parámetros establecidos en la configuración de IPtables.

Tabla 6.21.: Prueba unitaria PRU_020.

PRU_020	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado de la tabla raw de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectBuilder.
Método	buildRaw.
Precondiciones	-
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye el objeto Raw con los parámetros establecidos en la configuración de IPtables.

Tabla 6.22.: Prueba unitaria PRU_021.

PRU_021	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectUpdater.
Método	changePolicy.
Precondiciones	El objeto a actualizar debe existir.
Entrada	Contexto de servlet de la aplicación y datos de modificación de la tabla en cuestión.
Salida	El sistema modifica el objeto Filter, Nat, Mangle o Raw con los nuevos valores de los parámetros.

Tabla 6.23.: Prueba unitaria PRU_022.

PRU_022	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectUpdater.
Método	addRule.
Precondiciones	El objeto a actualizar debe existir.
Entrada	Contexto de servlet de la aplicación y datos de modificación de la tabla en cuestión.
Salida	El sistema modifica el objeto Filter, Nat, Mangle o Raw añadiendo la nueva regla especificada en la entrada.

Tabla 6.24.: Prueba unitaria PRU_023.

PRU_023	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectUpdater.
Método	deleteRule.
Precondiciones	El objeto a actualizar debe existir.
Entrada	Contexto de servlet de la aplicación y datos de modificación de la tabla en cuestión.
Salida	El sistema modifica el objeto Filter, Nat, Mangle o Raw eliminando la regla especificada en la entrada.

Tabla 6.25.: Prueba unitaria PRU_024.

PRU_024	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectUpdater.
Método	addChain.
Precondiciones	El objeto a actualizar debe existir.
Entrada	Contexto de servlet de la aplicación y datos de modificación de la tabla en cuestión.
Salida	El sistema modifica el objeto Filter, Nat, Mangle o Raw añadiendo la cadena especificada en la entrada.

Tabla 6.26.: Prueba unitaria PRU_025.

PRU_025	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectUpdater.
Método	renameChain.
Precondiciones	El objeto a actualizar debe existir.
Entrada	Contexto de servlet de la aplicación y datos de modificación de la tabla en cuestión.
Salida	El sistema modifica el objeto Filter, Nat, Mangle o Raw renombrando la cadena especificada en la entrada.

Tabla 6.27.: Prueba unitaria PRU_026.

PRU_026	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las tablas de la aplicación IPtables.
Capa	Servicios.
Clase	TableObjectUpdater.
Método	deleteChain.
Precondiciones	El objeto a actualizar debe existir.
Entrada	Contexto de servlet de la aplicación y datos de modificación de la tabla en cuestión.
Salida	El sistema modifica el objeto Filter, Nat, Mangle o Raw eliminando la cadena especificada en la entrada.

Tabla 6.28.: Prueba unitaria PRU_027.

PRU_027	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado del preprocesador Frag3 de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectBuilder.
Método	buildFrag3.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores frag3_global y frag3_engine.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye los objetos Frag3Global y Frag3Engine con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.29.: Prueba unitaria PRU_028.

PRU_028	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado del preprocesador Stream5 de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectBuilder.
Método	buildStream5.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores stream5_global, stream5_tcp, stream5_udp, stream5_icmp y stream5_ip.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye los objetos Stream5Global, Stream5TCP, Stream5UDP, Stream5ICMP, Stream5IP con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.30.: Prueba unitaria PRU_029.

PRU_029	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado del preprocesador SfPortscan de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectBuilder.
Método	buildSfPortscan.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentra definido el preprocesador sfportscan.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye los objetos SfPortscan con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.31.: Prueba unitaria PRU_030.

PRU_030	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado del preprocesador HTTPInspect de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectBuilder.
Método	buildHTTPInspect.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores http_inspect y http_inspect_server.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye los objetos HTTPInspectGlobal y HTTPInspectServer con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.32.: Prueba unitaria PRU_031.

PRU_031	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado del preprocesador Normalizer de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectBuilder.
Método	buildNormalizer.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores normalize_ip4 y normalize_tcp.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye los objetos NormalizerIP4 y NormalizerTCP con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.33.: Prueba unitaria PRU_032.

PRU_032	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado del preprocesador Reputation de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectBuilder.
Método	buildReputation.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores reputation.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye los objetos Reputation con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.34.: Prueba unitaria PRU_033.

PRU_033	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado del preprocesador Frag3 de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectUpdater.
Métodos	updateFrag3Global y updateFrag3Engine.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores frag3_global y frag3_engine. 3. En el sistema ya se encuentran construidos los objetos Frag3Global y Frag3Engine.
Entrada	Contexto de servlet de la aplicación y datos de modificación del preprocesador en cuestión.
Salida	1. El sistema actualiza los objetos Frag3Global y Frag3Engine con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizados los preprocesadores frag3_global y frag3_engine.

Tabla 6.35.: Prueba unitaria PRU_034.

PRU_034	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado del preprocesador Stream5 de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectUpdater.
Métodos	updateStream5Global, updateStream5TCP, Stream5UDP, Stream5-ICMP y updateStream5IP.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores stream5_global, stream5_tcp, stream5_udp, stream5_icmp y stream5_ip. 3. En el sistema ya se encuentran construidos los objetos Stream5-Global, Stream5TCP, Stream5UDP, Stream5ICMP y Stream5IP.
Entrada	Contexto de servlet de la aplicación y datos de modificación del preprocesador en cuestión.
Salida	1. El sistema actualiza los objetos Stream5Global, Stream5TCP, Stream5UDP, Stream5ICMP y Stream5IP con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizados los preprocesadores stream5_global, stream5_tcp, stream5_udp, stream5_icmp y stream5_ip.

Tabla 6.36.: Prueba unitaria PRU_035.

PRU_035	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado del preprocesador SfPortscan de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectUpdater.
Método	updateSfPortscan.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentra definido el preprocesador sfportscan. 3. En el sistema ya se encuentra construido el objeto SfPortscan.
Entrada	Contexto de servlet de la aplicación y datos de modificación del preprocesador en cuestión.
Salida	1. El sistema actualiza los objetos SfPortscan con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentra actualizado el preprocesador sfportscan.

Tabla 6.37.: Prueba unitaria PRU_036.

PRU_036	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado del preprocesador HTTPInspect de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectUpdater.
Método	updateHTTPInspectGlobal y updateHTTPInspectServer.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores http_inspect y http_inspect_server. 3. En el sistema ya se encuentran construidos los objetos HTTPInspectGlobal y HTTPInspectServer.
Entrada	Contexto de servlet de la aplicación y datos de modificación del preprocesador en cuestión.
Salida	1. El sistema actualiza los objetos HTTPInspectGlobal y HTTPInspectServer con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizados los preprocesadores http_inspect y http_inspect_server.

Tabla 6.38.: Prueba unitaria PRU_037.

PRU_037	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado del preprocesador Normalizer de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectUpdater.
Método	updateNormalizerIP4 y updateNormalizerTCP.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores normalize_ip4 y normalize_tcp. 3. En el sistema ya se encuentran construidos los objetos NormalizerIP4 y NormalizerTCP.
Entrada	Contexto de servlet de la aplicación y datos de modificación del preprocesador en cuestión.
Salida	1. El sistema actualiza los objetos NormalizerIP4 y NormalizerTCP con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizados los preprocesadores normalize_ip4 y normalize_tcp.

Tabla 6.39.: Prueba unitaria PRU_038.

PRU_038	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado del preprocesador Reputation de la aplicación Snort.
Capa	Servicios.
Clase	PreprocessorObjectUpdater.
Método	updateReputation.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidos los preprocesadores reputation. 3. En el sistema ya se encuentra construido el objeto Reputation.
Entrada	Contexto de servlet de la aplicación y datos de modificación del preprocesador en cuestión.
Salida	1. El sistema actualiza los objetos Reputation con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentra actualizado el preprocesador reputation.

Tabla 6.40.: Prueba unitaria PRU_039.

PRU_039	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado de las variables IP de la aplicación Snort.
Capa	Servicios.
Clase	VariableObjectBuilder.
Método	buildIPVariables.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas las variables IP.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye el objeto que almacena las variables IP con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.41.: Prueba unitaria PRU_040.

PRU_040	
Objetivo	Comprobar que el sistema es capaz de construir los objetos necesarios para el almacenamiento del estado de las variables de puertos de la aplicación Snort.
Capa	Servicios.
Clase	VariableObjectBuilder.
Método	buildIPVariables.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas las variables de puertos.
Entrada	Contexto de servlet de la aplicación.
Salida	El sistema construye el objeto que almacena las variables de puertos con los parámetros establecidos en el fichero de configuración de Snort.

Tabla 6.42.: Prueba unitaria PRU_041.

PRU_041	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las variables IP de la aplicación Snort.
Capa	Servicios.
Clase	VariableObjectUpdater.
Método	updateIPVariables.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas las variables IP. 3. En el sistema ya se encuentra construido el objeto que almacena las variables IP.
Entrada	Contexto de servlet de la aplicación y datos de modificación de las variables IP.
Salida	1. El sistema actualiza el objeto que almacena las variables IP con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizadas las variables IP.

Tabla 6.43.: Prueba unitaria PRU_042.

PRU_042	
Objetivo	Comprobar que el sistema es capaz de actualizar los objetos necesarios para el almacenamiento del estado de las variables de puertos de la aplicación Snort.
Capa	Servicios.
Clase	VariableObjectUpdater.
Método	updatePortVariables.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas las variables de puertos. 3. En el sistema ya se encuentra construido el objeto que almacena las variables de puertos.
Entrada	Contexto de servlet de la aplicación y datos de modificación de las variables de puertos.
Salida	1. El sistema actualiza el objeto que almacena las variables de puertos con los parámetros establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizadas las variables de puertos.

6.1.3. Pruebas de integración

La documentación de las pruebas incluidas dentro del nivel de pruebas de integración se realizará a través de los siguientes atributos:

- Identificador. Cada prueba dispondrá de un identificador único que la represente de manera unívoca. Este atributo será PRI_XXX, resultando necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar la prueba unitaria.
- Objetivos. Este atributo describirá los fines de la prueba a realizar.
- Capas. Este atributo identificará qué capas de la arquitectura están implicadas en la prueba en cuestión. Este atributo podrá tomar los siguientes valores:
  - Modelo.
  - Vista.
  - Controlador.
  - Servicios.
- Componentes. Este atributo identificará qué componentes del sistema están implicados en la prueba en cuestión. Este atributo podrá tomar como valor cualquiera de los nombres de los componentes obtenidos durante el diseño del sistema.
- Precondiciones. Este atributo identificará las condiciones que se tienen que cumplir para que sea posible realizar la prueba.
- Postcondiciones. Este atributo identificará las condiciones que se deben cumplir tras haber realizado la prueba.
- Entrada. Este atributo identificará el valor que se proporcionará a la prueba para comprobar el comportamiento del componente.
- Salida. Este atributo identificará el valor, o conjunto de valores, válido que el componente debe devolver para comprobar el comportamiento del mismo.

A continuación, se muestra un ejemplo de la manera en la que serán documentados las pruebas de integración.

Tabla 6.44.: Ejemplo de documentación de una prueba de integración.

PRI_000	
Objetivos	Mostrar un ejemplo de cómo documentar las pruebas de integración.
Capas	Modelo, Vista, Controlador.
Componentes	Módulo IPtables, Módulo Snort.
Precondiciones	-
Postcondiciones	Las pruebas de integración deben documentarse siguiendo este formato.
Entrada	Todas las pruebas unitarias deben seguir este formato.
Salida	Redactar los requisitos de usuario en este formato.

Identificación de pruebas de integración

Tabla 6.45.: Prueba de integración PRI_001.

PRI_001	
Objetivos	1. Comprobar que el sistema lanza correctamente, desde la interfaz web, la ejecución de la aplicación Snort y que el sistema accede correctamente al estado de ejecución de dicha aplicación. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente.
Precondiciones	La aplicación Snort no debe encontrarse en ejecución.
Postcondiciones	La aplicación Snort debe encontrarse en ejecución.
Entrada	Operación a realizar, en este caso, ejecución de la aplicación Snort.
Salida	La interfaz muestra el mensaje que indica que Snort se encuentra en ejecución.

Tabla 6.46.: Prueba de integración PRI_002.

PRI_002	
Objetivos	1. Comprobar que el sistema reinicia correctamente, desde la interfaz web, la ejecución de la aplicación Snort y que el sistema accede correctamente al estado de ejecución de dicha aplicación. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente.
Precondiciones	La aplicación Snort debe encontrarse en ejecución.
Postcondiciones	La aplicación Snort debe encontrarse en ejecución.
Entrada	Operación a realizar, en este caso, reinicio de la aplicación Snort.
Salida	La interfaz muestra el mensaje que indica que Snort se encuentra en ejecución.

Tabla 6.47.: Prueba de integración PRI_003.

PRI_003	
Objetivos	1. Comprobar que el sistema almacena correctamente el estado de los preprocesadores de la aplicación Snort. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas los preprocesadores Frag3, Stream5, SfPortscan, HTTPInspect, Normalizer y Reputation.
Postcondiciones	-
Entrada	Operación a realizar, en este caso, visualizar los preprocesadores de Snort.
Salida	Los valores de los parámetros de configuración ofrecidos por la interfaz del sistema se corresponden con los especificados en el fichero de configuración de Snort.

Tabla 6.48.: Prueba de integración PRI_004.

PRI_004	
Objetivos	1. Comprobar que el sistema actualiza correctamente el estado de los preprocesadores de la aplicación Snort. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas los preprocesadores Frag3, Stream5, SfPortscan, HTTPInspect, Normalizer y Reputation. 3. En el sistema ya se encuentra construido los objetos que almacenan los preprocesadores que van a ser actualizados.
Postcondiciones	1. El sistema actualiza los objetos que almacenan los parámetros de los preprocesadores con los valores establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizados los preprocesadores modificados.
Entrada	Operación a realizar, en este caso, actualizar los preprocesadores de Snort y los nuevos valores de los parámetros a modificar.
Salida	Los valores de los parámetros de configuración ofrecidos por la interfaz del sistema se corresponden con los especificados en la entrada del sistema.

Tabla 6.49.: Prueba de integración PRI_005.

PRI_005	
Objetivos	1. Comprobar que el sistema almacena correctamente el estado de las variables de la aplicación Snort. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas variables IP y variables de puertos.
Postcondiciones	-
Entrada	Operación a realizar, en este caso, visualizar las variables de Snort.
Salida	Los valores de los parámetros de configuración ofrecidos por la interfaz del sistema se corresponden con los especificados en el fichero de configuración de Snort.

Tabla 6.50.: Prueba de integración PRI_006.

PRI_006	
Objetivos	1. Comprobar que el sistema actualiza correctamente el estado de las variables de la aplicación Snort. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta al fichero de configuración de Snort. 2. En el fichero de configuración de Snort se encuentran definidas variables IP y variables de puertos. 3. En el sistema ya se encuentra construido los objetos que almacenan las variables que van a ser actualizadas.
Postcondiciones	1. El sistema actualiza los objetos que almacenan las variables de Snort con los valores establecidos en la entrada. 2. En el fichero de configuración de Snort se encuentran actualizadas las variables modificadas.
Entrada	Operación a realizar, en este caso, actualizar las variables de Snort.
Salida	Los valores de las variables de configuración ofrecidos por la interfaz del sistema se corresponden con los especificados en la entrada del sistema.

Tabla 6.51.: Prueba de integración PRI_007.

PRI_007	
Objetivos	1. Comprobar que el sistema visualiza correctamente las reglas de detección definidas en la aplicación Snort. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente.
Precondiciones	En la aplicación Snort se han especificado reglas de detección.
Postcondiciones	-
Entrada	Operación a realizar, en este caso, visualizar las reglas de Snort.
Salida	Las reglas ofrecidas por la interfaz del sistema coinciden con las que Snort está aplicando para detectar posibles intrusiones.

Tabla 6.52.: Prueba de integración PRI_008.

PRI_008	
Objetivos	Comprobar que el sistema visualiza correctamente las alertas generadas por la aplicación Snort.
Capas	Módulo, Vista, Servicios.
Componentes	Módulo Snort de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	En la configuración del sistema se ha especificado la ruta a la aplicación BASE.
Postcondiciones	-
Entrada	Operación a realizar, en este caso, visualizar las alertas de Snort.
Salida	Se muestra la interfaz de la aplicación BASE, la cual ofrece toda su funcionalidad al usuario.

Tabla 6.53.: Prueba de integración PRI_009.

PRI_009	
Objetivos	1. Comprobar que el sistema almacena correctamente el estado de las tablas de la aplicación IPtables. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo IPtables de las capas mencionadas anteriormente.
Precondiciones	-
Postcondiciones	-
Entrada	Operación a realizar, en este caso, visualizar el contenido de las tablas de IPtables.
Salida	El contenido de las tablas ofrecido por la interfaz del sistema se corresponde con el que almacena la aplicación IPtables.

Tabla 6.54.: Prueba de integración PRI_010.

PRI_010	
Objetivos	1. Comprobar que el sistema actualiza correctamente el estado de las tablas de la aplicación IPtables. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo IPtables de las capas mencionadas anteriormente.
Precondiciones	-
Postcondiciones	El sistema actualiza los objetos que almacenan las tablas de IPtables con los valores establecidos en la entrada.
Entrada	Operación a realizar, en este caso, actualizar el contenido de las tablas de IPtables.
Salida	Los valores de las tablas de IPtables ofrecidos por la interfaz del sistema se corresponden con los especificados en la entrada del sistema.

Tabla 6.55.: Prueba de integración PRI_011.

PRI_011	
Objetivos	1. Comprobar que el sistema guarda correctamente el estado de las tablas de la aplicación IPtables. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo IPtables de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	En la configuración del sistema se ha especificado la ruta en la que se almacenará el estado de la aplicación IPtables.
Postcondiciones	El sistema almacena el estado de la aplicación IPtables en el fichero especificado en la configuración del sistema.
Entrada	Operación a realizar, en este caso, guardar el estado de la aplicación IPtables.
Salida	El sistema muestra un mensaje en el que informa que la última fecha y hora de guardado del estado de la aplicación IPtables son la fecha y hora actuales.

Tabla 6.56.: Prueba de integración PRI_012.

PRI_012	
Objetivos	1. Comprobar que el sistema restaura correctamente el estado de las tablas de la aplicación IPtables. 2. Comprobar que la capa Controlador lleva a cabo su función correctamente.
Capas	Módulo, Vista, Controlador, Servicios.
Componentes	Módulo IPtables de las capas mencionadas anteriormente y Módulo de configuración.
Precondiciones	1. En la configuración del sistema se ha especificado la ruta en la que se encuentra almacenado el estado de la aplicación IPtables. 2. El estado de la aplicación ha sido guardado previamente.
Postcondiciones	El sistema restaura el estado de la aplicación IPtables que se encuentra en el fichero especificado en la configuración del sistema.
Entrada	Operación a realizar, en este caso, restaurar el estado de la aplicación IPtables.
Salida	Los valores de las tablas de IPtables ofrecidos por la interfaz del sistema se corresponden con los especificados en el estado guardado previamente.

6.1.4. Pruebas de sistema

La documentación de las pruebas incluidas dentro del nivel de pruebas de sistema se realizará a través de los siguientes atributos:

- Identificador. Cada prueba dispondrá de un identificador único que la represente de manera unívoca. Este atributo será PRS_XXX, resultando necesario sustituir XXX por un número de tres cifras, el cual será empleado para identificar la prueba unitaria.
- Descripción. Este atributo ofrecerá una breve descripción de la prueba a realizar.
- Precondiciones. Este atributo identificará las condiciones que se tienen que cumplir para que sea posible realizar la prueba.
- Entrada. Este atributo identificará el valor que se proporcionará a la prueba para comprobar el comportamiento del componente.
- Salida. Este atributo identificará el valor, o conjunto de valores, válido que el componente debe devolver para comprobar el comportamiento del mismo.

A continuación, se muestra un ejemplo de la manera en la que serán documentados las pruebas de sistema.

Tabla 6.57.: Ejemplo de documentación de una prueba de sistema.

PRS_000	
Descripción	Mostrar un ejemplo de cómo documentar las pruebas de sistema.
Capas	Modelo, Vista, Controlador.
Precondiciones	-
Entrada	Todas las pruebas de sistema deben seguir este formato.
Salida	Redactar los requisitos de usuario en este formato.

Identificación de pruebas de sistema

Tabla 6.58.: Prueba de sistema PRS_001.

PRS_001	
Descripción	Comprobar que es posible la conexión al sistema desde el computador en el que se encuentra instalado el mismo.
Capas	Modelo, Vista, Controlador, Servicios.
Precondiciones	1. El sistema se encuentra instalado. 2. El sistema ha sido configurado adecuadamente.
Entrada	Introducir en el navegador web la dirección en la que se encuentra el sistema.
Salida	En el navegador se muestra la página de inicio del sistema.

Tabla 6.59.: Prueba de sistema PRS_002.

PRS_002	
Descripción	Comprobar que es posible la conexión al sistema desde un computador que se encuentra en la misma red que el computador en el que se encuentra instalado el mismo.
Capas	Modelo, Vista, Controlador, Servicios.
Precondiciones	1. El sistema se encuentra instalado. 2. El sistema ha sido configurado adecuadamente.
Entrada	Introducir en el navegador web la dirección en la que se encuentra el sistema.
Salida	En el navegador se muestra la página de inicio del sistema.

Tabla 6.60.: Prueba de sistema PRS_003.

PRS_003	
Descripción	Comprobar que es posible la conexión al sistema desde un computador que no se encuentra en la misma red que el computador en el que se encuentra instalado el mismo.
Capas	Modelo, Vista, Controlador, Servicios.
Precondiciones	1. El sistema se encuentra instalado. 2. El sistema ha sido configurado adecuadamente.
Entrada	Introducir en el navegador web la dirección en la que se encuentra el sistema.
Salida	En el navegador se muestra la página de inicio del sistema.

Tabla 6.61.: Prueba de sistema PRS_004.

PRS_004	
Descripción	Comprobar que es posible el uso del sistema desde el computador en el que se encuentra instalado el mismo.
Capas	Modelo, Vista, Controlador, Servicios.
Precondiciones	1. El sistema se encuentra instalado. 2. El sistema ha sido configurado adecuadamente.
Entrada	Introducir en el navegador web la dirección en la que se encuentra el sistema.
Salida	Los parámetros de configuración de las aplicaciones IPtables y Snort coinciden con los especificados por el usuario.

Tabla 6.62.: Prueba de sistema PRS_005.

PRS_005	
Descripción	Comprobar que es posible el uso del sistema desde un computador que se encuentra en la misma red que el computador en el que se encuentra instalado el mismo.
Capas	Modelo, Vista, Controlador, Servicios.
Precondiciones	1. El sistema se encuentra instalado. 2. El sistema ha sido configurado adecuadamente.
Entrada	Introducir en el navegador web la dirección en la que se encuentra el sistema.
Salida	Los parámetros de configuración de las aplicaciones IPtables y Snort coinciden con los especificados por el usuario.

Tabla 6.63.: Prueba de sistema PRS_006.

PRS_006	
Descripción	Comprobar que es posible el uso del sistema desde un computador que no se encuentra en la misma red que el computador en el que se encuentra instalado el mismo.
Capas	Modelo, Vista, Controlador, Servicios.
Precondiciones	1. El sistema se encuentra instalado. 2. El sistema ha sido configurado adecuadamente.
Entrada	Introducir en el navegador web la dirección en la que se encuentra el sistema.
Salida	Los parámetros de configuración de las aplicaciones IPtables y Snort coinciden con los especificados por el usuario.

6.2. Análisis de consistencia

El objetivo de esta sección es comprobar que las pruebas establecidas en la sección anterior cubren los requisitos especificados para el sistema a desarrollar, y comprobar que todos ellos han sido probados.

Para cumplir este objetivo, se facilita la siguiente tabla, en la que se recoge por cada una de las pruebas cuáles de lo requisitos del sistema quedan probados.

Tabla 6.64.: Análisis de la consistencia pruebas-requisitos (Parte I).

	RS_001	RS_002	RS_003	RS_004	RS_005	RS_006	RS_007	RS_008
PRU_001		x						
PRU_002		x						
PRU_003		x						
PRU_004		x						
PRU_005			x					
PRU_006			x					
PRU_007			x					
PRU_008			x					
PRU_009			x					
PRU_010			x					
PRU_011			x					
PRU_012			x					
PRU_013			x					
PRU_014			x					
PRU_015								
PRU_016	x							
PRU_017			x					
PRU_018			x					
PRU_019			x					
PRU_020			x					
PRU_021			x					
PRU_022			x					
PRU_023			x					
PRU_024			x					
PRU_025			x					
PRU_026			x					
PRU_027		x				x		x
PRU_028		x				x		x
PRU_029		x				x		x

Tabla 6.65.: Análisis de la consistencia pruebas-requisitos (Parte II).

	RS_009	RS_010	RS_011	RS_012	RS_013	RS_014	RS_015	RS_016
PRU_001			x					
PRU_002				x				
PRU_003			x	x				
PRU_004			x	x				
PRU_005					x	x		
PRU_006					x			x
PRU_007					x		x	x
PRU_008					x		x	x
PRU_009					x		x	x
PRU_010					x	x		
PRU_011					x	x		
PRU_012					x	x		
PRU_013								
PRU_014								
PRU_015			x	x		x	x	x
PRU_016								
PRU_017					x			x
PRU_018					x			x
PRU_019					x			x
PRU_020					x			x
PRU_021					x	x	x	
PRU_022					x	x	x	
PRU_023					x	x	x	
PRU_024					x	x	x	
PRU_025					x	x	x	
PRU_026					x	x	x	
PRU_027								
PRU_028								
PRU_029								

Tabla 6.66.: Análisis de la consistencia pruebas-requisitos (Parte III).

	RS_017	RS_018	RS_019	RS_020	RS_021	RS_022	RS_023	RS_024
PRU_001								
PRU_002								
PRU_003								
PRU_004								
PRU_005		x		x				
PRU_006	x							
PRU_007	x	x		x				
PRU_008	x	x		x				
PRU_009	x	x		x				
PRU_010		x		x				
PRU_011		x		x				
PRU_012		x		x				
PRU_013			x					
PRU_014			x	x	x	x		
PRU_015	x		x	x	x	x		
PRU_016								
PRU_017	x							
PRU_018	x							
PRU_019	x							
PRU_020	x							
PRU_021		x		x				
PRU_022		x		x				
PRU_023		x		x				
PRU_024		x		x				
PRU_025		x		x				
PRU_026		x		x				
PRU_027								
PRU_028								
PRU_029								

Tabla 6.67.: Análisis de la consistencia pruebas-requisitos (Parte IV).

	RS_001	RS_002	RS_003	RS_004	RS_005	RS_006	RS_007	RS_008
PRU_030		x				x		x
PRU_031		x				x		x
PRU_032		x				x		x
PRU_033		x				x		x
PRU_034		x				x		x
PRU_035		x				x		x
PRU_036		x				x		x
PRU_037		x				x		x
PRU_038		x				x		x
PRU_039		x			x		x	
PRU_040		x			x		x	
PRU_041		x			x		x	
PRU_042		x			x		x	
PRI_001		x						
PRI_002		x						
PRI_003		x				x		
PRI_004		x				x		x
PRI_005		x			x			
PRI_006		x			x		x	
PRI_007		x						
PRI_008		x		x				
PRI_009			x					
PRI_010			x					
PRI_011			x					
PRI_012			x					
PRS_001								
PRS_002								
PRS_003								
PRS_004								
PRS_005								
PRS_006								

Tabla 6.68.: Análisis de la consistencia pruebas-requisitos (Parte V).

	RS_009	RS_010	RS_011	RS_012	RS_013	RS_014	RS_015	RS_016
PRU_030								
PRU_031								
PRU_032								
PRU_033								
PRU_034								
PRU_035								
PRU_036								
PRU_037								
PRU_038								
PRU_039								
PRU_040								
PRU_041								
PRU_042								
PRI_001			x					
PRI_002				x				
PRI_003								
PRI_004								
PRI_005								
PRI_006								
PRI_007	x	x						
PRI_008								
PRI_009					x			
PRI_010					x	x	x	
PRI_011								
PRI_012								
PRS_001								
PRS_002								
PRS_003								
PRS_004								
PRS_005								
PRS_006								

Tabla 6.69.: Análisis de la consistencia pruebas-requisitos (Parte VI).

	RS_017	RS_018	RS_019	RS_020	RS_021	RS_022	RS_023	RS_024
PRU_030								
PRU_031								
PRU_032								
PRU_033								
PRU_034								
PRU_035								
PRU_036								
PRU_037								
PRU_038								
PRU_039								
PRU_040								
PRU_041								
PRU_042								
PRI_001								
PRI_002								
PRI_003								
PRI_004							x	x
PRI_005								
PRI_006							x	x
PRI_007								
PRI_008								
PRI_009								
PRI_010		x					x	x
PRI_011			x	x				
PRI_012					x	x		
PRS_001								
PRS_002								
PRS_003								
PRS_004								
PRS_005								
PRS_006								

Observando estas tablas, se puede concluir que todos los requisitos quedan probados a través del plan de pruebas que se ha diseñado, por lo que una vez llevado a cabo el mismo se podrá dar por concluida la fase de implementación del sistema.

6.3. Evaluación del sistema

En esta sección se desarrollará una evaluación del sistema final, para verificar el correcto funcionamiento del mismo. Esta evaluación debe ser realizada tras haber finalizado la etapa de implementación del sistema, puesto que servirá para analizar si realmente se han alcanzado los objetivos que se pretendían al comienzo de este trabajo y para detectar posibles errores pasados por alto durante las fases de análisis y diseño del sistema.

Además, dicha evaluación debe ser llevada a cabo por una o varias personas que cuenten con el perfil de usuario final de la aplicación (el cual fue expuesto en la sección 4.1). Los motivos principales de esta decisión son los que se listan a continuación:

- Recrear lo más fielmente posible el entorno real en el que operará la aplicación.
- El usuario podrá informar al equipo de desarrollo de su punto de vista acerca de la aplicación, cuáles le parecen que son los puntos fuertes, qué partes de la aplicación modificaría o eliminaría, o qué funcionalidad añadiría a la aplicación. Esta información puede resultar muy útil a la hora de realizar futuras actualizaciones del sistema.
- El usuario puede informar al equipo de desarrollo acerca de si la aplicación resulta sencilla de utilizar, y en caso de no ser así, ofrecer nuevas ideas para mejorar la misma.
- Resulta posible detectar cuáles son los errores más comunes a la hora de realizar las tareas por parte de los usuarios y mejorar la aplicación en esos puntos más "problemáticos".

El desarrollo del proceso de evaluación será entregar la versión desarrollada del sistema a una serie de potenciales usuarios de la aplicación, explicándoles para qué sirve la herramienta y las tareas que deben realizar. A su vez, se pedirá a estos usuarios que expresen su parecer acerca de la aplicación que están usando. Tras llevar a cabo este proceso será posible analizar los datos recogidos para evaluar finalmente la aplicación.

Tal y como venimos exponiendo, desde prácticamente el inicio de este documento, el fin que se persigue con el desarrollo de esta aplicación es facilitar el trabajo de los administradores de seguridad de sistemas Linux. Por tanto, para una evaluación lo más fiel posible a la carga de trabajo con la que contará la aplicación trataremos de simular, en la medida de lo posible, el trabajo de estas personas. Para ello se van a establecer una secuencia de tareas, propias de administradores de seguridad, como son la configuración de algunos de los parámetros de Snort e IPtables para detectar o evitar ciertas acciones sobre el computador que defienden. Dichas tareas se llevarán a cabo tanto de forma tradicional, es decir, a través de comandos de consola y ficheros de configuración, como con la ayuda de la aplicación desarrollada. El motivo de esta decisión es poder desarrollar una comparativa entre ambas soluciones. De este modo podremos evaluar si el diseño de la aplicación es correcto en el sentido en que los usuarios de la misma la encuentran de utilidad a la hora de llevar a cabo su trabajo.

Por otra parte, tras haber llevado a cabo esta configuración a través de la aplicación desarrollada, resulta necesario ponerse en la piel de un posible atacante cuyo objetivo es realizar las acciones que el administrador de seguridad intenta evitar en el computador que gestiona. De esta forma será posible verificar que el diseño del sistema realmente es funcional a la hora de realizar dicha configuración y cumple con la tarea para la que fue diseñada.

Como las tareas que se pueden realizar a través de la configuración de Snort e IPtables son prácticamente infinitas, debemos acotar cuáles de ellas serán empleadas para la realización de la evaluación. Por simplificar un poco este proceso y no extender demasiado la documentación del mismo, las tareas que se proponen para evaluar el sistema son las siguientes:

1. Configurar la aplicación Snort para registrar cualquier intento de conocer si el computador en el que se encuentra instalada la aplicación se encuentra conectado.

2. Comprobar que la configuración establecida en el paso anterior ha sido recogida en el sistema.

3. Comprobar que la aplicación Snort se encuentra en estado de ejecución. Tanto si no se encuentra en estado de ejecución como si se encuentra en este estado, será necesario lanzar la ejecución de dicha aplicación. En el primer caso para que pueda comenzar a realizar la detección de intrusiones, y en el segundo caso para que pueda aplicar los cambios realizados en el paso anterior.

4. Tras realizar esta configuración se enviará un mensaje ICMP al computador en el que se encuentra instalado el sistema desarrollado.

5. Visualizar las alertas generadas por Snort desde la aplicación a través de la aplicación BASE. Puesto que en el paso anterior se envió un mensaje para conocer si el computador en el que se encuentra instalada la aplicación se encuentra conectado, Snort debe haber generado una alerta para este evento.

6. Configurar la aplicación IPtables para almacenar en los registros del sistema cualquier intento de conocer si el computador en el que se encuentra instalada la aplicación se encuentra conectado.

7. Tras realizar esta configuración, se enviará de nuevo un mensaje ICMP al computador en el que se encuentra instalado el sistema desarrollado.

8. Visualizar el registro del sistema operativo. Puesto que en el paso anterior se envió un mensaje para conocer si el computador en el que se encuentra instalada la aplicación se encuentra conectado, IPtables debe haber generado una entrada para este evento en dicho registro. Esta tarea tiene como objetivo comprobar que el sistema a configurado correctamente la aplicación IPtables, ya que la funcionalidad de visualizar el registro del sistema operativo no está incluida en el sistema desarrollado.

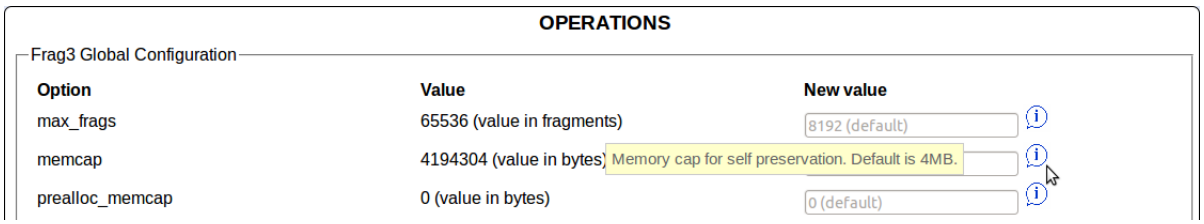

Figura 6.1.: Una de las descripciones emergentes que encontramos en la aplicación.

Pese a tratarse de unas tareas muy sencillas, resultan ser bastante representativas de lo que podría ser una serie de tareas que debe desarrollar un administrador de seguridad en sistemas Linux en el desarrollo de su trabajo.

6.3.1. Resultados

En esta sección se muestran los resultados obtenidos tras aplicar el sistema de evaluación diseñado en la sección anterior.

Tras facilitar la versión inicial del sistema a un grupo de usuarios potenciales del mismo, podemos decir que el sistema desarrollado parece contar con una buena acogida entre dichos usuarios. Estos usuarios destacan la facilidad en el aprendizaje de cómo usar la aplicación debido, principalmente, al estilo homogéneo mantenido a lo largo de toda la aplicación, lo cual hace que el usuario no precise memorizar cómo realizar cada tarea o cada tipo de tareas, sino que es capaz de razonar acerca de qué acciones precisa llevar a cabo para la consecución de cada tarea. Sin embargo, además de deberse a este motivo, también destacan la gran utilidad de la diversa información de ayuda que ofrece la aplicación, en particular, la utilidad de las descripciones emergentes que facilitan al usuario la modificación de los parámetros de seguridad y evitan la necesidad de consultar los largos manuales de usuario de cada aplicación.

Otro aspecto, también muy relevante, relacionado con este último, es que el sistema informa de que se han tratado de modificar o añadir uno o varios parámetros con un valor que no está permitido, destacando cuáles son los valores erróneos y no efectuando ningún cambio en el estado de la aplicación implicada hasta que no sean corregidos dichos valores.

Los usuarios que han realizado la evaluación también destacan la rapidez a la hora de llevar a cabo cualquier tarea: sólo resulta necesario seleccionar la herramienta de seguridad con la que trabajar y comprobar la configuración actual de sus parámetros o seleccionar qué elemento de la misma se desea actualizar. Con estas dos elecciones, el usuario ya tiene a su alcance las acciones que puede llevar a cabo. Y además, lo sencillo que resulta ejecutarla: basta con rellenar un simple formulario en el que se muestran los parámetros que se pueden configurar, los valores que tienen asignados, la entrada para cambiar dichos valores y las descripciones emergentes de cada parámetro.

La parte negativa del sistema, destacada por los usuarios es que, a la hora de realizar la primera de las tareas establecidas para llevar a cabo la evaluación, se encontraban algo perdidos y confusos. Esta confusión viene provocada porque el sistema desarrollado

no provee la funcionalidad necesaria para editar los ficheros de texto que contienen las reglas de detección de la aplicación Snort, sólo provee la funcionalidad necesaria para visualizar las reglas contenidas en dichos ficheros. Por tanto, para llevar a cabo esta tarea, el usuario debe conectarse al sistema en el que se encuentra instalado Snort y modificar, manualmente, el fichero de reglas de detección. Tras comprobar si podía haber existido algún error durante las fases de análisis o diseño del proyecto de nuestra parte, se comprendió que dicha funcionalidad no se había recogido para que formara parte del sistema a la hora de especificar cuáles debían ser las necesidades a cubrir. Por tanto, no se considera que haya tenido lugar ningún error durante la fase de desarrollo del sistema, ya que, simplemente la funcionalidad de gestionar las reglas de detección de Snort desde la aplicación quedaba fuera del alcance del proyecto. Esto se debió a la heterogeneidad y complejidad de dichas reglas de detección. No obstante, este aspecto formará, sin duda alguna, parte de las posibles mejoras y actualizaciones del sistema, puesto que los usuarios han echado en falta esta funcionalidad en el mismo.

Por otra parte, en lo que respecta a la funcionalidad del sistema, podemos decir que cumple con lo establecido en los capítulos anteriores y, por tanto, con los objetivos iniciales marcados para este proyecto. Podemos mantener esta postura puesto que las tareas establecidas en el proceso de evaluación del sistema, cuyo objetivo era verificar el correcto funcionamiento del mismo, se completaron con éxito.

Capítulo 7.

Conclusiones y líneas futuras

7.1. Conclusiones

A lo largo de este trabajo se ha construido una herramienta para ayudar a defender y monitorizar sistemas Linux, facilitando las tareas más comunes de los administradores de seguridad de estos sistemas. El desarrollo de la aplicación se ha realizado teniendo en consideración, principalmente, el diseño de la interfaz de usuario, con el fin de que ésta resulte lo más intuitiva y sencilla posible para el usuario. Como principal conclusión, puede decirse que todos los objetivos establecidos al inicio del desarrollo del trabajo han quedado satisfechos.

Para la construcción de esta herramienta se ha comenzado estudiando cuáles son los principales mecanismos de seguridad para defender sistemas Linux, destacando entre ellos los cortafuegos y los sistemas de detección de intrusiones. Este estudio también incluía el análisis de las herramientas más destacadas que implementan cada mecanismo, siendo IPtables y Snort las encargadas de implementar los mecanismos mencionados anteriormente. Este estudio ha ayudado a tener una mejor visión acerca de cuál es el funcionamiento de estos mecanismos.

Posteriormente se han analizado otras herramientas que permiten la gestión y monitorización de la actividad generada por IPtables y Snort, concluyendo que la mayor parte de ellas podían ser fácilmente mejoradas para prestar más y mejor funcionalidad al usuario final.

Una de las cuestiones más complicadas a la que nos hemos enfrentado se ha presentado durante la planificación del sistema. Esta complicación se debe a que resulta difícil estimar el tiempo que será necesario dedicar a cada fase del desarrollo. Esto se vio agravado debido a que en dicha planificación había que tener en cuenta el tiempo que sería necesario dedicar a cada asignatura del curso, las cuales se estaban cursando durante el desarrollo de este trabajo. Esta dificultad a la hora de planificar el trabajo se hace evidente tras ser necesario emplear tres meses de trabajo más de los que fueron planificados. Sin embargo, esto resultará de ayuda en el desarrollo de futuros proyectos, puesto que servirá como experiencia, algo con lo que prácticamente no se contaba al comenzar este trabajo.

Pese a que uno de los principales motivos por los que fue escogida la plataforma de desarrollo JEE para la implementación de la herramienta fue la experiencia adquirida con ella a lo largo de la carrera, ha demostrado que ofrece muchas ventajas. Entre ellas

se pueden destacar: la modularidad con la que permite implementar la aplicación, la gran utilidad de mecanismos orientados a la presentación web y la gran utilidad de mecanismos orientados a la recepción de peticiones de los usuarios. En esta elección hubo que descartar otras tecnologías, como Grails, con las que hubiera resultado muy interesante trabajar.

Durante el proceso de implementación de la herramienta, se han llevado a cabo las pruebas necesarias para garantizar que la misma cuenta con la funcionalidad necesaria para la que ha sido diseñada y su funcionamiento es el esperado. También ha sido probado el comportamiento de la herramienta, comprobando que es el adecuado frente a distintas situaciones. Después de este proceso de implementación y pruebas, se realizó un proceso de evaluación de la herramienta llevado a cabo por parte de un grupo de usuarios potenciales de la aplicación, para conocer sus impresiones acerca de la misma, y recoger posibles aspectos de mejora de la misma.

A pesar de tratarse de una aplicación bastante sencilla, durante su desarrollo ha sido necesario atajar ciertos problemas. Estos problemas aparecieron, principalmente, durante la fase de implementación del sistema, sobretodo a la hora de recuperar la información de cada preprocesador de Snort desde el fichero de configuración de la herramienta. Esto se debió a que cada preprocesador cuenta con un formato distinto a la hora de almacenar tanto cada parámetro de configuración, como cada valor de estos parámetros. Por tanto, automatizar la tarea de recuperar cada uno de estos valores ha sido algo tedioso y complicado. También surgieron problemas a la hora de desarrollar la funcionalidad necesaria para ejecutar comandos externos a la herramienta desarrollada (con el fin de poder interactuar con las herramientas IPtables y Snort). Esto se debió a dos motivos: el primero es que nunca antes había sido programada una funcionalidad similar, y el segundo es que la documentación acerca de cómo abarcar este problema es escasa y poco útil.

Tras completar el desarrollo de la herramienta, se puede concluir que la funcionalidad de ésta es bastante completa y que cubre todas las necesidades expresadas en los requisitos de usuario recogidos en la sección 4.2. Sin embargo, la herramienta ofrece muchas mejoras que podrán ser llevadas a cabo en el futuro, algunas de las cuales serán propuestas más adelante. De esta forma cualquier usuario interesado podrá tener acceso libre al código fuente de la herramienta y llevar a cabo las mejoras propuestas para la misma, o cualquier otra mejora que, probablemente, no se encuentre entre las propuestas en este documento. Por otra parte, cualquier usuario puede beneficiarse de las ventajas que esta herramienta ofrece.

7.2. Líneas futuras

A continuación se propone una serie de líneas futuras para la ampliación de la funcionalidad de la herramienta desarrollada, y que así ésta ofrezca más apoyo a los administradores de seguridad de sistemas Linux.

- Incluir la funcionalidad necesaria para que sea posible gestionar las reglas de detección de la herramienta Snort. Esta funcionalidad permitiría llevar a cabo las siguientes tareas:
  - Añadir, modificar o eliminar reglas de detección.
  - Comprobar que, al añadir una regla, la sintaxis de la misma es correcta.
  - Crear nuevos ficheros de reglas.
  - Seleccionar qué ficheros de reglas se incluyen en la configuración de la herramienta.
- Incluir la funcionalidad necesaria para que sea posible gestionar más tipos de variables y preprocesadores de Snort.
- Incluir la funcionalidad necesaria para que sea posible gestionar las opciones con las que la herramienta Snort es ejecutada.
- Incluir la funcionalidad necesaria para que sea posible monitorizar la actividad generada por la herramienta IPtables. Esta monitorización se llevaría a cabo accediendo al registro del sistema y recuperando aquellos mensajes contenidos en el mismo que hayan sido generados por dicha herramienta.
- Incluir la funcionalidad necesaria para que sea posible visualizar las reglas de cada tabla de la herramienta IPtables organizadas según los criterios especificados por el usuario (por ejemplo, sólo visualizar aquellas reglas relacionadas con un protocolo completo, sólo visualizar aquellas reglas que cuya acción sea incluir un mensaje en el registro del sistema operativo, etc.).
- Incluir la funcionalidad necesaria para que sea posible gestionar y monitorizar la actividad de otras herramientas de seguridad estudiadas al comienzo de este documento, así como de otras que no fueron documentadas entonces.
- Incluir la funcionalidad necesaria para que sea posible llevar a cabo un control de acceso en la herramienta.
- Incluir la funcionalidad necesaria para que sea posible configurar los parámetros de la aplicación a través de la interfaz de usuario de la misma.
- Incluir más idiomas en los que mostrar la información presentada por la herramienta ya que, actualmente, sólo se encuentra disponible en inglés.

- Incluir la funcionalidad necesaria para que sea posible emplear la herramienta en otros sistemas operativos ya que, actualmente, sólo se encuentra disponible para Ubuntu 10.10 o superior.
- Realizar un estudio sobre la simplicidad y usabilidad de la interfaz de usuario de la aplicación, y proponer mejoras para mejorar dichos aspectos de la interfaz.
- Optimizar la herramienta para que sea posible acceder a la misma desde otros navegadores ya que, actualmente, sólo se encuentra optimizada para Mozilla Firefox 14.0 o superior.
- Desarrollar vídeotutoriales sobre cómo se utiliza la aplicación.

Llevar a cabo todas estas mejoras no resultaría complicado debido a la modularidad con la que ha sido diseñada e implementada la herramienta, por lo que cualquier usuario interesado en mejorar la funcionalidad de la misma puede sentirse libre de modificar los componentes necesarios para implementar estas u otras mejoras ideadas por ellos mismos que no se encuentren recogidos en los puntos anteriores.

Bibliografía

[1] Subrahmanyam Allamaraju. *Programación Java Server con J2EE Edición 1.3*. Anaya Multimedia, 2008.

[2] Julia H. Allen. *The CERT Guide to System and Network Security Practices*. Addison-Wesley Professional, 2001.

[3] Ross Anderson. *Security engineering: a guide to building dependable distributed systems*. Wiley, 2008.

[4] Oskar Andreasson. Documento electrónico: Tutorial de iptables 1.1.19es, Mayo 2008. URL `http://ccia.ei.uvigo.es/docencia/SSI/practicas/iptables-tutorial.es.pdf`.

[5] Stig Saether Bakken. *Manual de PHP*. Rafael Martínez, 2001.

[6] Proyecto BASE. Página web: Basic analysis and security engine (base), 2012. URL `http://base.secureideas.net/index.php`.

[7] Nacho Brito Calahorro. *Manual de desarrollo web con GRAILS*. ImaginaWorks, 2009.

[8] Roman Danyliw. Página web: Analysis console for intrusion databases (acid), 2003. URL `http://acidlab.sourceforge.net/`.

[9] Escuela de Groovy. Página web: Escuela de groovy, 2011. URL `http://www.escueladegroovy.com/informacion/groovy-grails`.

[10] Jefatura del Estado. Ley orgÁnica 15/1999, de 13 de diciembre,de protección de datos de carácter personal. *Boletín Oficial del Estado*, 1999. URL `http://www.boe.es/boe/dias/1999/12/14/pdfs/A43088-43099.pdf`.

[11] doc ubuntu.es. Documentación de ubuntu: Iptables, 2012. URL `http://doc.ubuntu-es.org/Iptables`.

[12] Proyecto Fedora. Tutorial: Filtrado de paquetes en redes con iptables, 2009. URL `http://proyectofedora.org/wiki/Tutorial_de_filtrado_de_paquetes_en_redes_con_iptables#Tablas`.

[13] Luis Miguel Cabezas Granado. *Manual imprescindible de PHP5*. Anaya, 2004.

[14] Gene Guinter. Página web: Proyecto snez, 2012. URL `http://geneguinter.com/`.

[15] Antonio Villalón Huerta. *Seguridad en UNIX y Redes*. -, 2002.

[16] IBM. Página web: Ibm - españa, 2011. URL `http://www-142.ibm.com/software/products/es/es/swarchitect-websphere/`.

[17] James F. Kurose. *Redes de computadoras: un enfoque descendente*. Pearson Education, 2010.

[18] Rami Lehti. Página web: Advanced intrusion detection environment (aide), 2011. URL `http://aide.sourceforge.net/`.

[19] Theo Mandel. *The Elements of User Interface Design*. Wiley, 1997.

[20] William Metcalf. Página web: Snort inline. URL `http://snort-inline.sourceforge.net/oldhome.html`.

[21] Pierpaolo Palazzoli. Documento electrónico: Utilizando snort inline, 2006. URL `http://snortattack.org/docs/SNORT_ES.pdf`.

[22] Pablo Suau Pérez. Página web: Proyecto phpiptables, 2002. URL `http://www.alu.ua.es/p/psp4/`.

[23] Robert Richardson. 2010 / 2011 csi computer crime and security survey, Diciembre 2010. URL `https://cours.etsmtl.ca/log619/documents/divers/CSIsurvey2010.pdf`.

[24] Karen Scarfone. Guide to intrusion detection and prevention systems (idps). *Recommendations of the National Institute of Standards and Technology*, Febrero 2007.

[25] Proyecto SNORT. Documento electrónico: Snort users manual, 2011. URL `http://www.snort.org/assets/166/snort_manual.pdf`.

[26] Proyecto SNORT. Página web: Snort, 2012. URL `http://www.snort.org/`.

[27] Sourceforge. Página web: Proyecto php firewall generator, 2009. URL `http://sourceforge.net/projects/phpfwgen/`.

[28] TIOBE. Página web: Tiobe programming community index for july 2012, 2012. URL `http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html`.

Apéndice A.

Manual de usuario

A.1. Introducción

El objetivo de este apéndice es facilitar un manual de la aplicación desarrollada, a través del cual el usuario final pueda:

- Conocer qué requisitos debe cumplir un computador para poder instalar y hacer uso de la aplicación. Estos requisitos mínimos han sido recogidos en la sección A.2 de este manual.
- Llevar a cabo la instalación del software necesario para el correcto funcionamiento de la aplicación. Esta guía de instalación ha sido recogida en la sección A.3.1 de este manual.
- Llevar a cabo la instalación de la propia aplicación. Esta guía de instalación ha sido recogida en la sección A.3.2 de este manual.
- Llevar a cabo la configuración del entorno operacional necesaria para el correcto funcionamiento de la aplicación. Esta guía de configuración ha sido recogida en la sección A.4.1 de este manual.
- Llevar a cabo la configuración de la propia aplicación. Esta guía de configuración ha sido recogida en la sección A.4.2 de este manual.
- Comenzar a tomar contacto con la aplicación, y conocer cómo se llevan a cabo algunas de las tareas de seguridad más comunes a través de la misma. Esta guía de uso de la aplicación ha sido recogido en la sección A.5 de este manual.

A.2. Requisitos mínimos del sistema

Para poder hacer uso de la aplicación, el usuario debe disponer de un computador en el que se encuentre instalado, como mínimo, los siguientes requisitos:

- Sistema operativo Ubuntu 10.10 o superior.
- Java Enterprise Edition 6.
- Servidor web Apache Tomcat 6.0.35 o superior.
- IPtables 1.4.4 o superior.
- Snort 2.9.2 o superior.
- BASE.
- Conexión a Internet.

Además de estos requisitos, es necesario disponer de un navegador web Mozilla Firefox 14.0.1 o superior, a través del cual resulte posible acceder a la interfaz de usuario de la aplicación. Sin embargo, este software no tiene porqué encontrarse instalado en el mismo computador en el que ejecute la aplicación, puesto que será posible acceder a la misma desde cualquier otro computador que disponga del navegador mencionado y conexión a Internet.

A.3. Instalación

A.3.1. Preparación del entorno de trabajo

Instalación de IPtables

Para el correcto funcionamiento de la aplicación es requisito imprescindible que en el sistema se encuentre instalada la herramienta IPtables versión 1.4.4 o superior. Todas las distribuciones Ubuntu del sistema operativo Linux que son compatibles con la aplicación desarrollada, se proporcionan con una versión del software IPtables, con la que la aplicación puede operar, ya preinstalada. Por lo tanto, basta con instalar una versión válida de la distribución Ubuntu para tener preparado el software IPtables.

Instalación de Snort y BASE

Para el correcto funcionamiento de la aplicación es requisito imprescindible que en el sistema se encuentren instaladas la herramientas Snort versión 2.9.2 o superior y BASE[1]. Para la correcta instalación de este software es necesario llevar a cabo los siguientes pasos:

1. Instalar en el sistema los paquetes de software que pueden ser necesarios para el funcionamiento de las herramientas Snort y BASE. La instalación de estos paquetes puede ser llevada a cabo a través del gestor de paquetes del sistema operativo, o bien directamente desde la consola del mismo. Los nombres de cada uno de los paquetes son los siguientes:

   a) apache2
   b) libapache2-mod-php5
   c) libwww-perl
   d) mysql-server
   e) mysql-common
   f) mysql-client
   g) php5-mysql
   h) libnet1
   i) libnet1-dev
   j) libpcre3
   k) libpcre3-dev
   l) autoconf
   m) libcrypt-ssleay-perl
   n) libmysqlclient-dev

[1] En este caso es indeferente la versión de la herramienta que se encuentre instalada.

ñ) php5-gd

o) php-pear

p) libphp-adodb

q) php5-cli

r) libtool

s) libssl-dev

t) gcc-4.4

u) g++

v) automake

w) make

x) flex

y) bison

2. Instalar la librería adodb. Para ello es necesario seguir estos pasos:

   a) Descargar la librería desde la dirección `http://sourceforge.net/projects/adodb/files/adodb-php-4-and-5/adodb-4992-for-php/` y descomprimir el fichero descargado dentro del directorio /usr/share/php.

3. Instalar la librería libpcap. Para ello es necesario seguir estos pasos:

   a) Descargar la librería desde la dirección `http://www.tcpdump.org/release/libpcap-1.1.1.tar.gz` y ubicar el fichero descargado dentro del directorio /usr/src/.

   b) En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd /usr/src
$ tar -zxvf libpcap-1.2.1.tar.gz
$ cd libpcap-1.2.1
$ ./configure -prefix=/usr -enable-share make && make install
```

4. Instalar la librería libdnet. Para ello es necesario seguir estos pasos:

   a) Descargar la librería desde la dirección `http://libdnet.googlecode.com/files/libdnet-1.12.tgz` y ubicar el fichero descargado dentro del directorio /usr/src/.

   b) En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd /usr/src
$ tar -zxvf libdnet-1.12.tgz
$ cd libdnet-1.12
$ ./configure -prefixx=/usr -enable-share && make && make install
```

5. Instalar la librería daq. Para ello es necesario seguir estos pasos:

   a) Descargar la librería desde la dirección `http://www.snort.org/downloads/1098` y ubicar el fichero descargado dentro del directorio /usr/src/.

   b) En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd /usr/src
$ tar -zxvf daq-0.6.2.tar.gz
$ cd daq-0.6.2
$ ./configure && make && make install
```

6. Actualizar la ruta a las librerías compartidas del sistema. Para ello es necesario introducir el siguiente comando en una consola del sistema operativo:

```
$ echo >> /etc/ld.so.conf /usr/lib && ldconfig
```

7. Instalar la herramienta Snort. Para ello es necesario seguir estos pasos:

   a) Descargar la herramienta desde la dirección `http://www.snort.org/snort-downloads` y ubicar el fichero descargado dentro del directorio /usr/src/.

   b) Descargar las reglas de detección desde la dirección `https://www.snort.org/snort-rules/` y ubicar los ficheros descargados dentro del directorio /usr/src/.

   c) En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd /usr/src
$ tar -zxvf snort-2.9.2.1.tar.gz
$ cd snort-2.9.2.1
$ ./configure -with-mysql -enable-dynamicplugin
-enable-perfprofiling -enable-ipv6 -enable-zlib
-enable-reload && make && make install

$ mkdir /etc/snort /etc/snort/rules /var/log/snort
/var/log/barnyard2 /usr/local/lib/snort_dynamicrules

$ groupadd snort
$ useradd -g snort snort

$ chown snort:snort /var/log/snort /var/log/barnyard2

$ cd /usr/src/snort-2.9.1/etc
$ cp *.conf* /etc/snort/
$ cp *.map* /etc/snort/

$ cd /usr/src && mkdir rules && cd rules
```

```
$ tar -zxvf snortrules-snapshot-2920.tar.gz
$ cp -r rules/ /etc/snort
```

8. Descargar el fichero de configuración de Snort, con el formato adecuado para que lo pueda manejar la aplicación, de la dirección `http://dl.dropbox.com/u/13710761/snort.conf`. Sustituir el fichero de configuración de Snort por el fichero descargado.

9. Editar el fichero de configuración de la herramienta Snort y llevar a cabo los siguientes cambios en él:

```
ipvar HOME_NET any (Red que deseamos proteger)
ipvar EXTERNAL_NET any
var RULE_PATH ./rules
...
# preprocessor normalize_ip4
# preprocessor normalize_tcp: ips ecn stream
# preprocessor normalize_icmp4
# preprocessor normalize_ip6
# preprocessor normalize_icmp6
```

10. Crear la base de datos que almacenará la información relativa a las alertas generadas por Snort. Para ello es necesario introducir los siguientes comandos en una consola del sistema operativo:

```
$ mysql -u root -p
mysql> create database snort;
mysql> grant CREATE, INSERT, SELECT, DELETE, UPDATE
on snort.* to snort@localhost;
mysql> SET PASSWORD FOR snort@localhost=PASSWORD('snort');
mysql> flush privileges;
mysql> exit;
$ mysql -u root -p < /usr/src/snort-2.9.2.1/schemas/create_mysql
snort
```

11. Instalar la herramienta barnyard2. Para ello es necesario seguir estos pasos:

    a) Descargar la librería desde la dirección `http://www.securixlive.com/download/barnyard2/barnyard2-1.9.tar.gz` y ubicar el fichero descargado dentro del directorio /usr/src/.

    b) En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd /usr/src
$ tar -zxf barnyard2-1.9.tar.gz
$ cd barnyard2-1.9
$ ./configure -with-mysql && make && make install
$ mv /usr/local/etc/barnyard2.conf /etc/snort
```

12. Editar el fichero de configuración de la herramienta barnyard2 y llevar a cabo los siguientes cambios en él:

```
#output alert_fast: stdout
output alert_fast
...
output database: log, mysql, user=snort password=snort dbname=snort
host=localhost
```

13. Instalar la herramienta BASE. Para ello es necesario seguir estos pasos:

    a) Descargar la librería desde la dirección `http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-1.4.5.tar.gz` y ubicar el fichero descargado dentro del directorio /usr/src/.

    b) En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd /usr/src
$ tar -zxf base-1.4.5.tar.gz
$ cp -r base-1.4.5 /var/www/base
$ chmod 777 /var/www/base
$ chmod 755 /var/www/base (Al terminar la instalación)
```

14. Editar el fichero de configuración de la herramienta BASE y llevar a cabo los siguientes cambios en él:

```
error_reporting = E_ALL & ~E_NOTICE
```

15. En una consola del sistema operativo introducir los siguientes comandos:

```
$ pear config-set preferred_state alpha
$ pear install Image_Color
$ pear install Image_Canvas-alpha
$ pear install Image_Graph-alpha

$ /etc/init.d/apache2 restart
```

16. Abrir un navegador web, introducir en la barra de dirección localhost/base y seguir los pasos que se indican.

Instalación de Apache Tomcat 6.0.35

Para el correcto funcionamiento de la aplicación es requisito imprescindible que en el sistema se encuentre instalado el servidor web Apache Tomcat versión 6.0.35 o superior. Para la correcta instalación de este software es necesario llevar a cabo los siguientes pasos:

1. Descargar el servidor web desde la dirección `http://tomcat.apache.org/download-60.cgi`. Existen varios formatos en los que descargar este software, pero para seguir el resto de pasos que se especifican en este manual será necesario descargar una distribución binaria en formato ZIP.

2. En una consola del sistema operativo introducir los siguientes comandos:

```
$ cd Descargas/
$ unzip apache-tomcat-6.0.35.zip
$ sudo mv apache-tomcat-6.0.35 /usr/local/tomcat
$ cd /usr/local/tomcat/bin
$ sudo chmod 755 catalina.sh
$ sudo chmod 755 digest.sh
$ sudo chmod 755 setclasspath.sh
$ sudo chmod 755 shutdown.sh
$ sudo chmod 755 startup.sh
$ sudo chmod 755 tool-wrapper.sh
$ sudo chmod 755 version.sh
```

3. Crear un fichero, de nombre "tomcat", en el directorio /etc/init.d. Editar dicho fichero e incluir las siguientes líneas:

```
# Script para el arranque del servidor Tomcat
# como un servicio del sistema operativo.

case $1 in
start)
    sh /usr/local/tomcat/bin/startup.sh
    ;;
stop)
    sh /usr/local/tomcat/bin/shutdown.sh
    ;;
restart)
    sh /usr/local/tomcat/bin/shutdown.sh
    sh /usr/local/tomcat/bin/startup.sh
    ;;
esac
exit 0
```

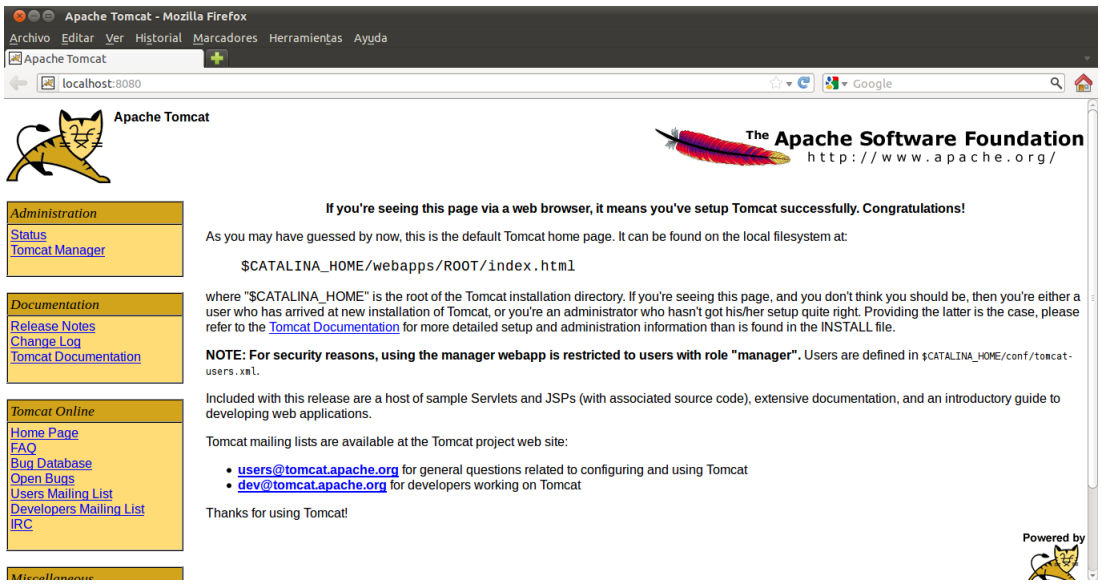

Figura A.1.: Página HTML para la gestión del servidor web Apache Tomcat 6.0.35.

4. Generar los enlaces simbólicos al fichero creado en el paso anterior en los directorios rc0.d, rc1.d, rc2.d, etc., para que el servidor web se inicie con el arranque del sistema operativo. Para ello es necesario introducir los siguientes comandos en una consola del sistema operativo:

```
$ sudo chmod 755 /etc/init.d/tomcat
$ sudo update-rc.d tomcat defaults
```

5. Editar el fichero /usr/local/tomcat/conf/tomcat-users.xml e incluir las siguientes líneas en el mismo:

```
<tomcat-users>
    <role rolename=''manager-gui''/>
    <user username=''tomcat'' password=''tomcat'' roles=''manager-gui''/>
</tomcat-users>
```

6. Arrancar la ejecución del servidor web. Para ello es necesario introducir el siguiente comando en una consola del sistema operativo:

```
$ sudo service tomcat start
```

7. Comprobar que el servidor web ha sido instalado correctamente y se encuentra en funcionamiento. Para ello es necesario arrancar un navegador web e introducir como dirección localhost:8080. Si todos los pasos anteriores se han llevado a cabo correctamente, el navegador debe mostrar la página HTML de la Figura (A.1).

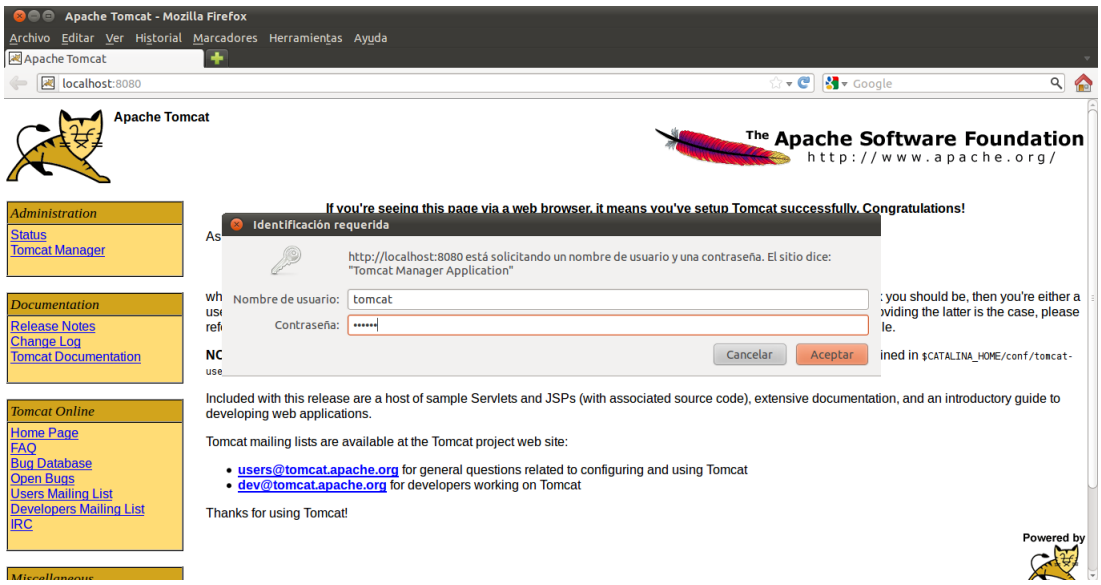

Figura A.2.: Control de acceso para la gestión del servidor web Apache Tomcat 6.0.35.

A.3.2. Instalación de la aplicación

Tras preparar el entorno de trabajo en el que trabajará la aplicación, es necesario instalar la misma. Para la correcta instalación de este software es necesario llevar a cabo los siguientes pasos:

1. Descargar la aplicación desde la dirección `http://dl.dropbox.com/u/13710761/WebApplication.war`.

2. Arrancar un navegador web e introducir como dirección localhost:8080 y acceder a "Tomcat Manager". En este punto, como se muestra en la Figura (A.2), la aplicación solicitará un usuario y contraseña. Estos datos de autenticación fueron especificados en el quinto paso de la sección Instalación de Apache Tomcat 6.0.35.

3. Ir a la sección "Desplegar", seleccionar como "Archivo WAR a desplegar" el fichero que ha sido descargado en el primer paso de esta sección y pulsar el botón "Desplegar", tal y como se muestra en la Figura.(A.3).

4. Comprobar que la aplicación ha sido instalada correctamente. Para ello es necesario arrancar un navegador web Mozilla Firefox 14.0.1 o superior e introducir como dirección localhost:8080/WebApplication. Si todos los pasos anteriores se han llevado a cabo correctamente, el navegador debe mostrar la página de bienvenida de la aplicación.

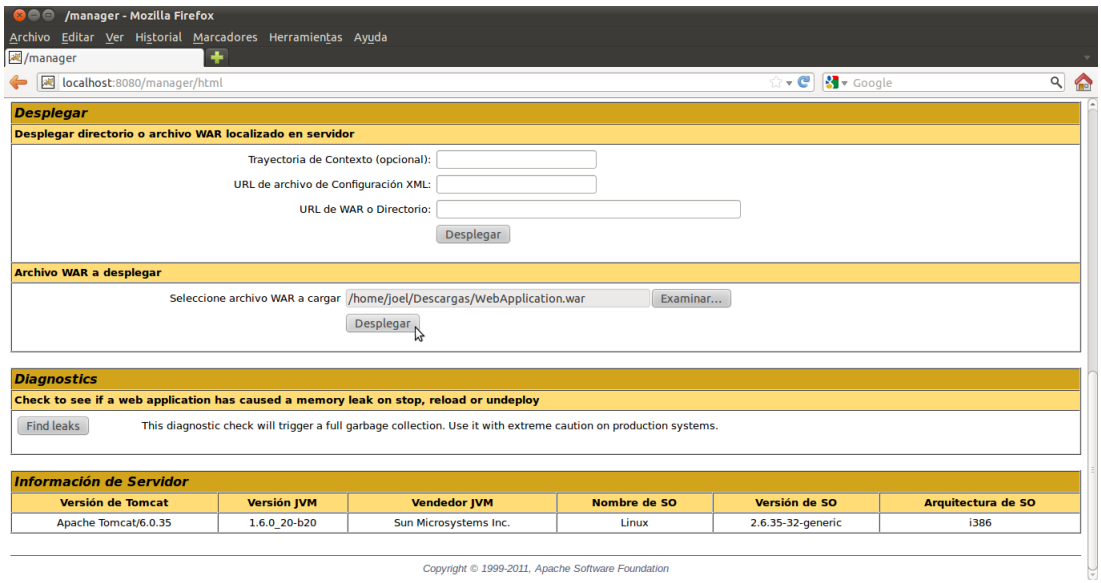

Figura A.3.: Despliegue de la aplicación.

A.4. Configuración

A.4.1. Configuración del entorno de trabajo

Para el correcto funcionamiento de la aplicación es necesario otorgar al usuario del sistema operativo, el cual se encargará de ejecutar la aplicación web, ciertos privilegios para ejecutar Snort e IPtables. Para ello será necesario seguir los siguientes pasos:

1. Editar el fichero /etc/sudoers, teniendo en cuenta que el usuario desde el que se lleva a cabo esta edición debe tener permiso para modificar dicho fichero. Para ello es necesario introducir el siguiente comando en una consola del sistema operativo:

   ```
   $ sudo visudo
   ```

2. Incluir las siguientes líneas en el fichero, teniendo en cuenta que "[usuario]" es el usuario a través del cual será ejecutada la aplicación.:

   ```
   [usuario] ALL=NOPASSWD:/usr/local/bin/snort, /usr/bin/pkill snort,
   /usr/local/bin/barnyard2, /usr/bin/pkill barnyard2
   [usuario] ALL=NOPASSWD:/sbin/iptables
   [usuario] ALL=NOPASSWD:/sbin/iptables-save
   [usuario] ALL=NOPASSWD:/sbin/iptables-restore
   ```

3. Guardar los cambios introducidos en el fichero y salir de la edición del mismo.

A.4.2. Configuración de la aplicación

Para llevar a cabo la configuración de la aplicación y poder hacer uso de todas las funcionalidades que ésta proporciona es necesario seguir los siguientes pasos:

1. Editar el fichero /usr/local/tomcat/webapps/WebApplication/WEB-INF /classes/jbm/tfg/utils/configuration/conf/conf.properties.

2. Establecer el valor adecuado para los siguientes parámetros:

   a) IP. A través de este parámetro se especifica la dirección IP del computador en el que se encuentra instalada la aplicación.

   b) snortConfFilePath. A través de este parámetro se especifica la ruta en la que se encuentra instalado el fichero de configuración de la herramienta Snort.

   c) snortRulesDirectoryPath. A través de este parámetro se especifica la ruta en la que se encuentra instalado el directorio que contiene las reglas de detección de Snort.

   d) BASE. A través de este parámetro se especifica la dirección a través de la cual se accede a la herramienta BASE.

e) iptablesState. A través de este parámetro se especifica la ruta en la que se almacenará el fichero que contendrá el estado guardado de la aplicación IP-tables.

Cada uno de estos parámetros cumplen con el patrón "NombreDelParámetro=Valor". De esta forma, para establecer el valor de cada uno de ellos sólo es necesario modificar la parte derecha de la expresión mostrada.

A.5. Gestión de las herramientas de seguridad

Para seleccionar la herramienta de seguridad que se desea gestionar, es suficiente con hacer click sobre el nombre de dicha herramienta, que aparece en el menú de la pantalla de bienvenida. En la Figura (A.4) se muestra la pantalla de bienvenida de la aplicación.

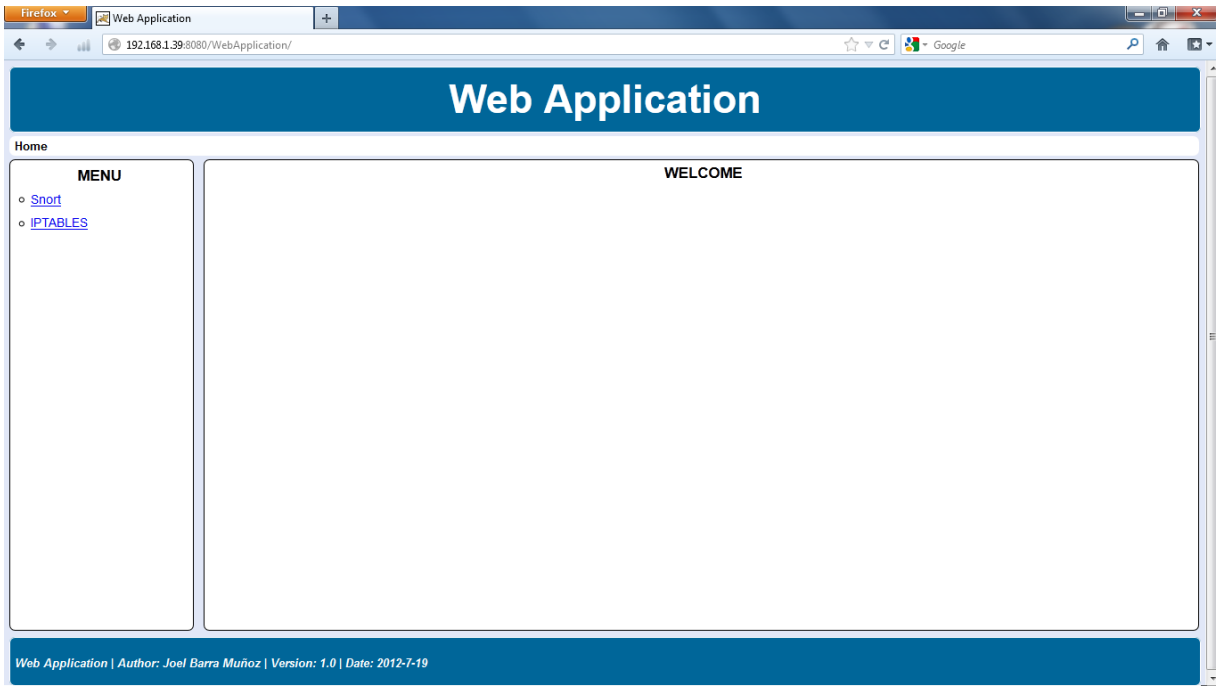

Figura A.4.: Pantalla de bienvenida de la aplicación.

Tras haber realizado esto, se mostrarán el resto de acciones que es posible llevar a cabo para la gestión de cada una de las aplicaciones. Estas acciones serán tratadas a continuación, siendo organizadas en función de la herramienta que permitan gestionar.

Es importante hacer notar que, a lo largo de todas las pantallas que conforman la aplicación, encontraremos la misma organización de los elementos presentados en ellas: las migas de pan siempre se encontrarán bajo el título principal de la aplicación, el menú con los parámetros que pueden ser configurados siempre se encontrará a la izquierda de la pantalla, y los elementos que permiten gestionar las herramientas de seguridad siempre se encontrarán a la derecha de la pantalla.

A.5.1. Gestión de Snort

Pantalla principal de la gestión de Snort

Al hacer click sobre "Snort" en la pantalla de bienvenida, nos encontramos con la pantalla que se muestra en la Figura (A.5). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Migas de pan. Este elemento presenta la parte de la aplicación en la que se encuentra actualmente el usuario, y permite volver a páginas anteriores haciendo click sobre el nombre de la página a la que se desea volver.

2. Menú. Este elemento presenta cada uno de los parámetros generales que pueden ser configurados o visualizados a través de la aplicación. Estos parámetros son: variables de configuración, preprocesadores, reglas de detección y acceso a la aplicación BASE.

3. Estado de Snort. A través de este elemento es posible visualizar el estado de ejecución de Snort (que podrá ser en ejecución "running", o parado "stopped"), así como lanzar o reiniciar la ejecución de dicha herramienta, lo cual puede llevarse a cabo haciendo clikc en el enlace "click here".

4. Información relevante acerca de algunos de los elementos de Snort que pueden ser configurados a través de la aplicación.

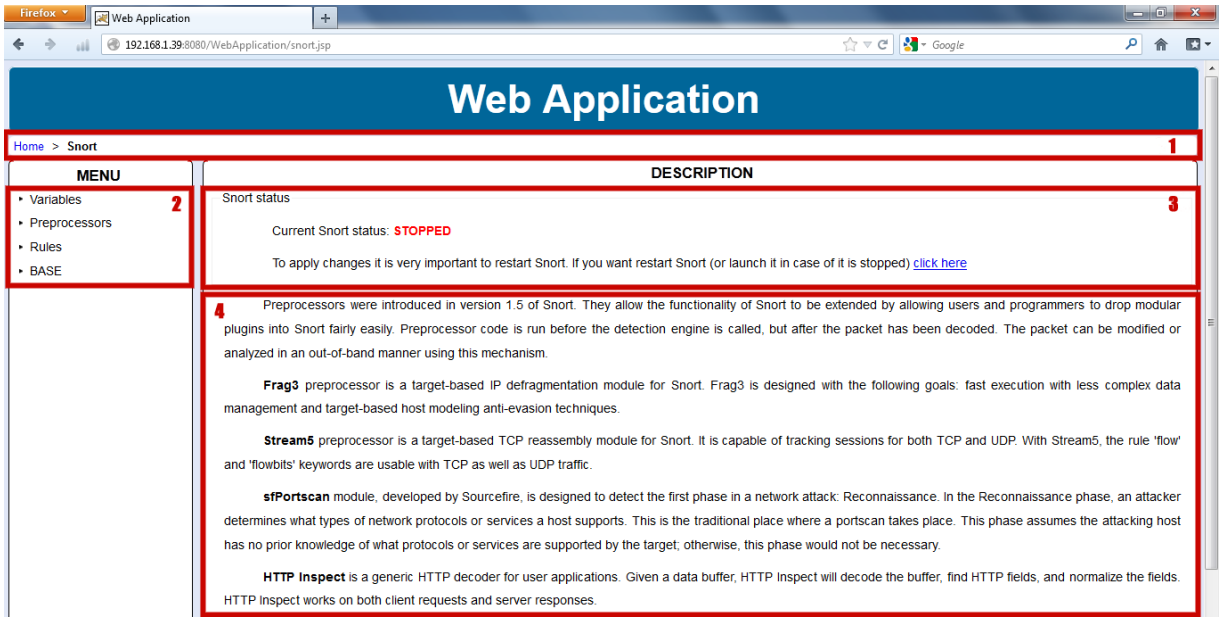

Figura A.5.: Pantalla principal de la gestión de Snort.

Gestión de las variables de configuración

Al hacer click sobre "Variables" en la pantalla principal de la gestión de Snort, se despliegan los tipos de variables que pueden ser manejadas a través de la aplicación.

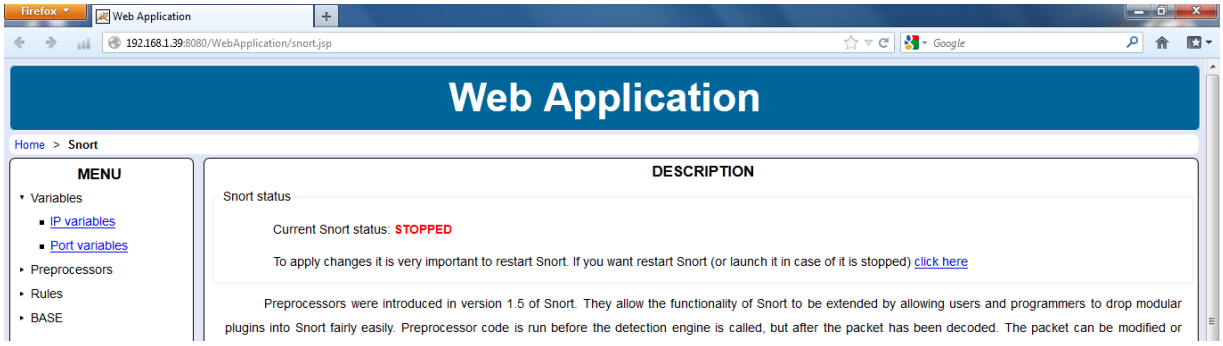

Figura A.6.: Variables que pueden ser gestionadas a través de la aplicación.

Al seleccionar uno de estos tipos se accede a la pantalla que se muestra en la Figura (A.7). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre y valor de cada una de las variables que han sido especificadas en la herramienta Snort.

2. Entrada para la actualización de las variables que han sido especificadas en la herramienta Snort.

3. Campos para especificar una variable nueva, junto con su valor correspondiente.

4. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

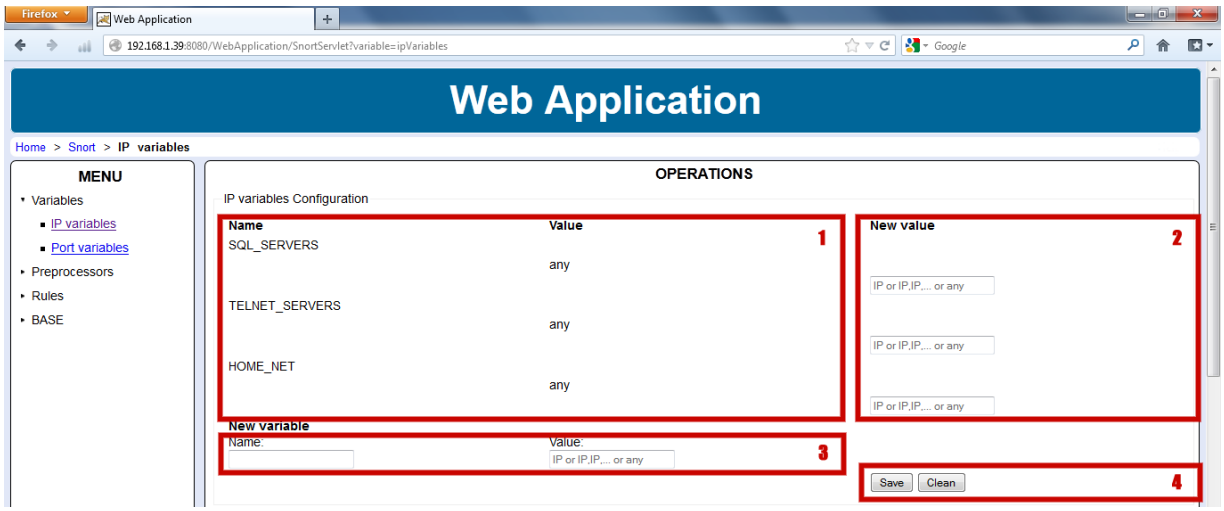

Figura A.7.: Pantalla para la gestión de las variables IP de Snort.

Gestión de los preprocesadores

Al hacer click sobre "Preprocessor" en la pantalla principal de la gestión de Snort, se despliegan los preprocesadores que pueden ser manejados a través de la aplicación.

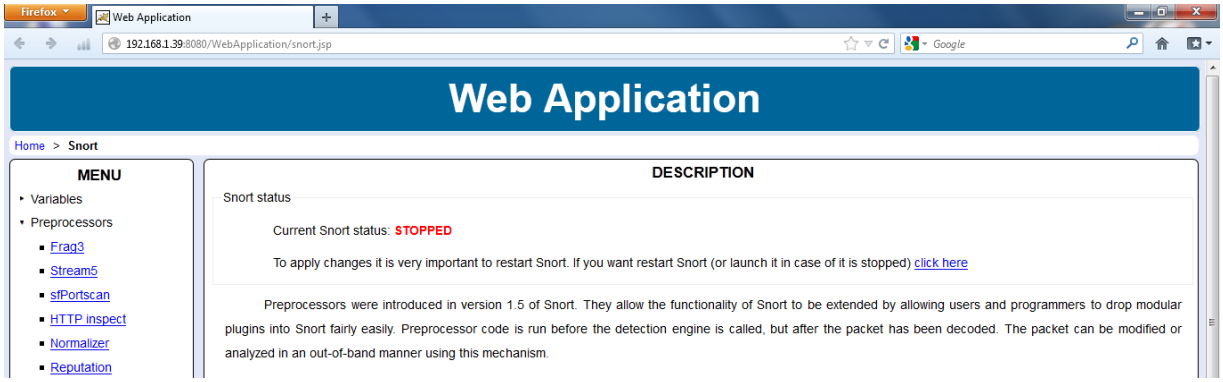

Figura A.8.: Preprocesadores que pueden ser gestionados a través de la aplicación.

Al seleccionar uno de estos preprocesadores se accede a la pantalla que se muestra en la Figura (A.9). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre del preprocesador que se está configurando.
2. Nombre y valor de cada uno de los parámetros del preprocesador que han sido especificados en la herramienta Snort.
3. Entrada para la actualización de los parámetros del preprocesador que han sido especificados en la herramienta Snort.
4. Descripción emergente con ayuda para la correcta configuración de cada parámetro.
5. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

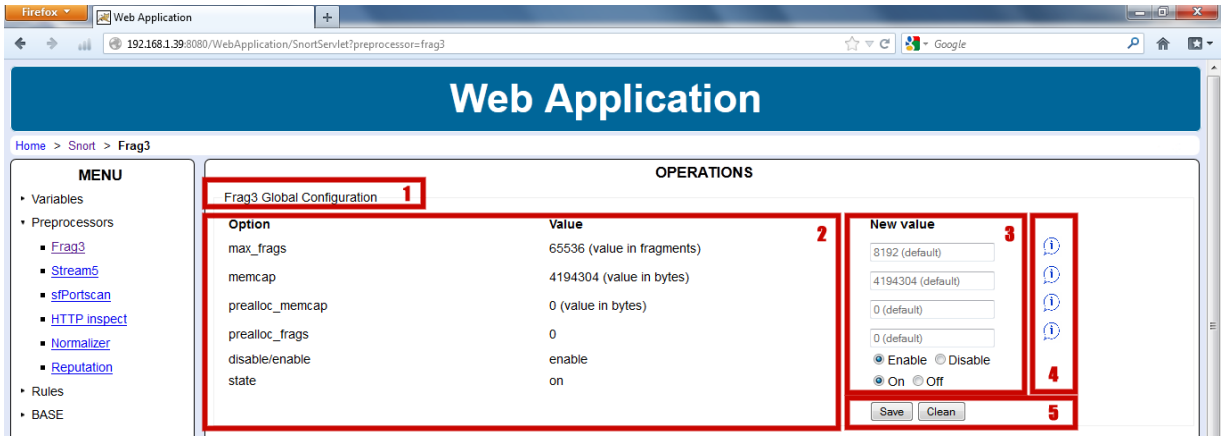

Figura A.9.: Pantalla para la gestión del preprocesador frag3 de Snort.

Visualización de las reglas de detección

Al hacer click sobre "Rules" en la pantalla principal de la gestión de Snort, se despliegan los nombres de los ficheros de reglas que pueden ser visualizados a través de la aplicación.

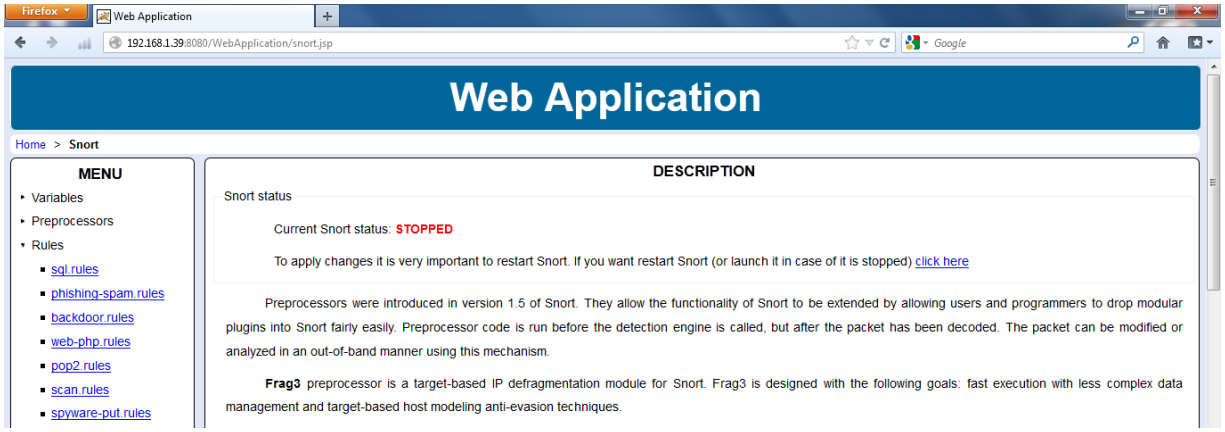

Figura A.10.: Ficheros de reglas que pueden ser visualizados a través de la aplicación.

Al seleccionar uno de estos ficheros de reglas se accede a la pantalla que se muestra en la Figura (A.11). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre del fichero de reglas que se está visualizando.

2. Cada una de las reglas contenidas en el fichero que se está visualizando.

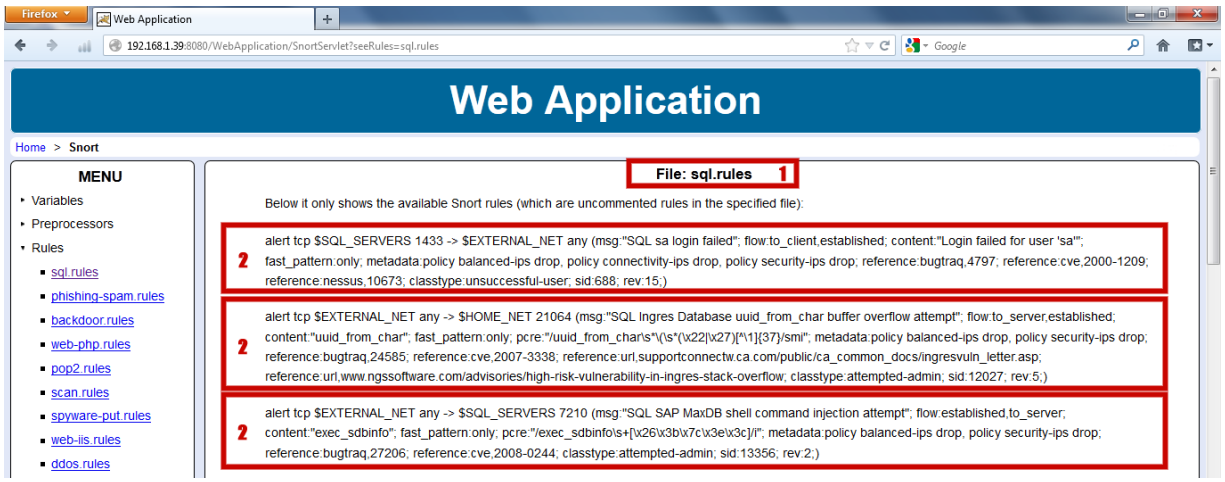

Figura A.11.: Pantalla para la visualización de las reglas de detección de Snort.

Visualización de las alertas generadas

Al hacer click sobre "BASE" en la pantalla principal de la gestión de Snort, se despliega el enlace para el acceso a la herramienta BASE.

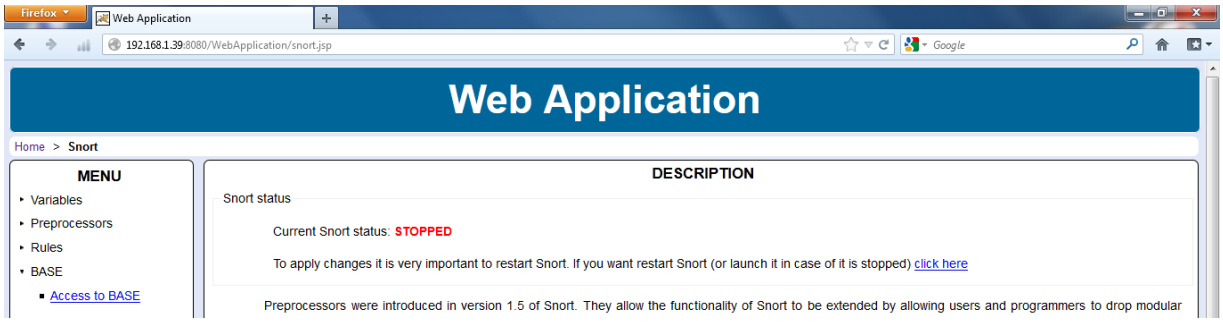

Figura A.12.: Enlace para el acceso a la herramienta BASE.

Al seleccionar este enlace, se despliega en el navegador la página de inicio de la herramienta BASE, tal y como se muestra en la Figura (A.13).

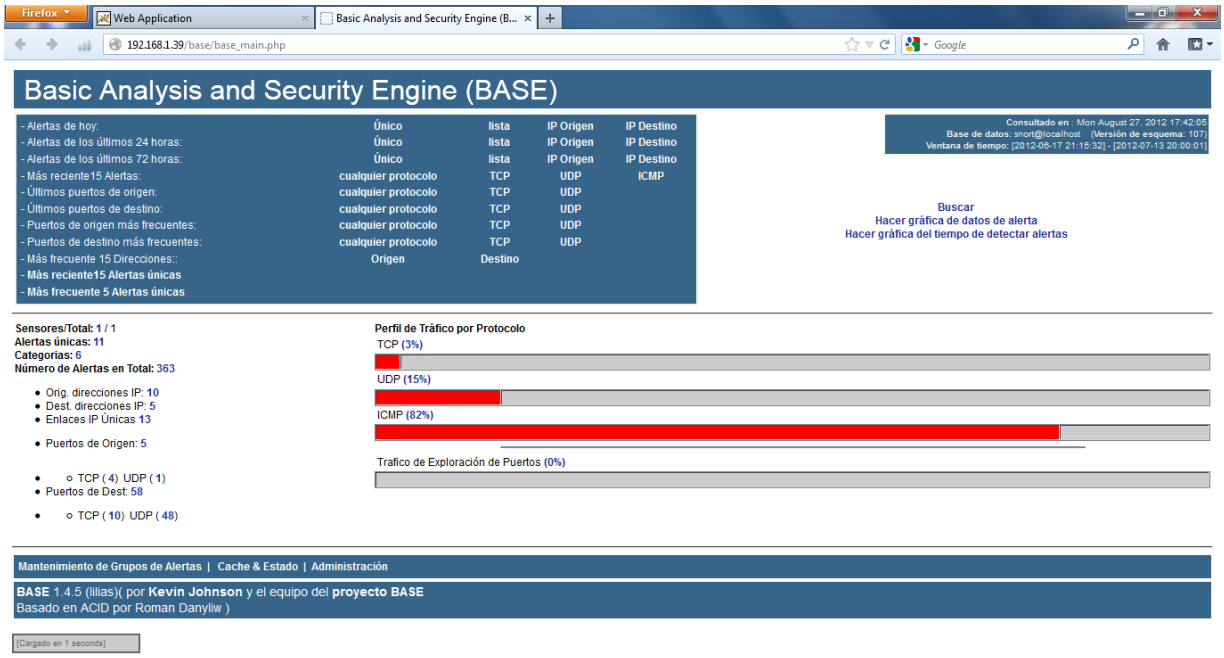

Figura A.13.: Pantalla de inicio de la herramienta BASE.

A.5.2. Gestión de IPtables

Pantalla principal de la gestión de IPtables

Al hacer click sobre "IPtables" en la pantalla de bienvenida, nos encontramos con la pantalla que se muestra en la Figura (A.14). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Migas de pan. Este elemento presenta la parte de la aplicación en la que se encuentra actualmente el usuario, y permite volver a páginas anteriores haciendo click sobre el nombre de la página a la que se desea volver.

2. Menú. Este elemento presenta cada una de las tablas que pueden ser gestionadas o visualizadas a través de la aplicación. Estas tablas son todas las que ofrece la aplicación IPtables: filter, nat, mangle y raw.

3. Información relevante acerca de algunos de los elementos de IPtables que pueden ser configurados a través de la aplicación.

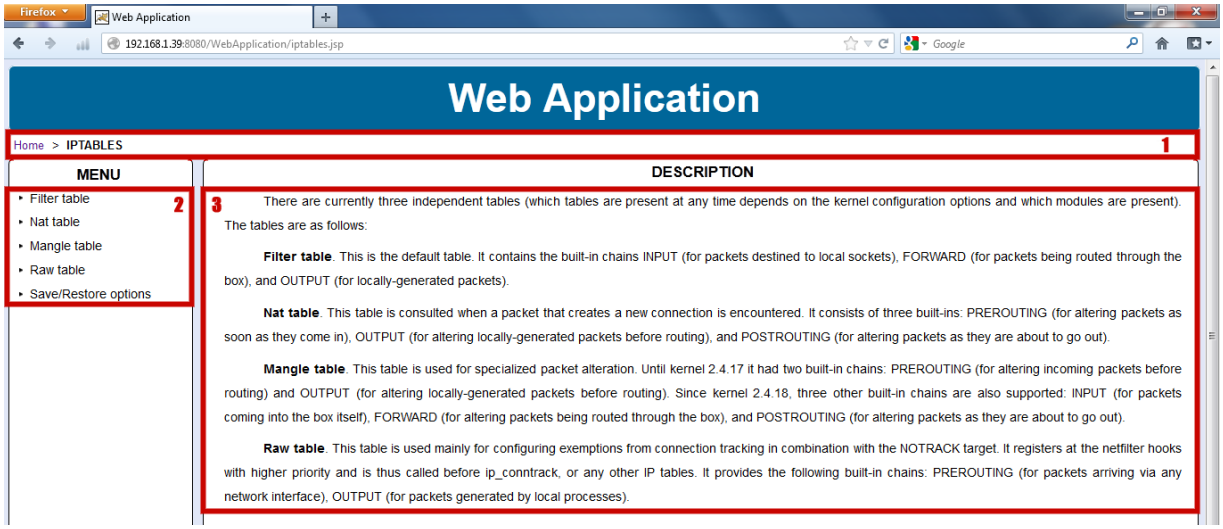

Figura A.14.: Pantalla principal de la gestión de IPtables.

Debido a que la gestión de cada una de las tablas se lleva a cabo a través de las mismas acciones, sólo se mostrará cómo llevar a cabo las tareas para una de las cuatro tablas disponibles. La gestión del resto de las tablas se lleva a cabo de manera análoga a la aquí explicada.

Cambio de la política por defecto de una cadena

Al hacer click sobre "Filter table" en la pantalla principal de la gestión de IPtables, se despliegan las acciones que pueden ser llevadas a cabo a través de la aplicación.

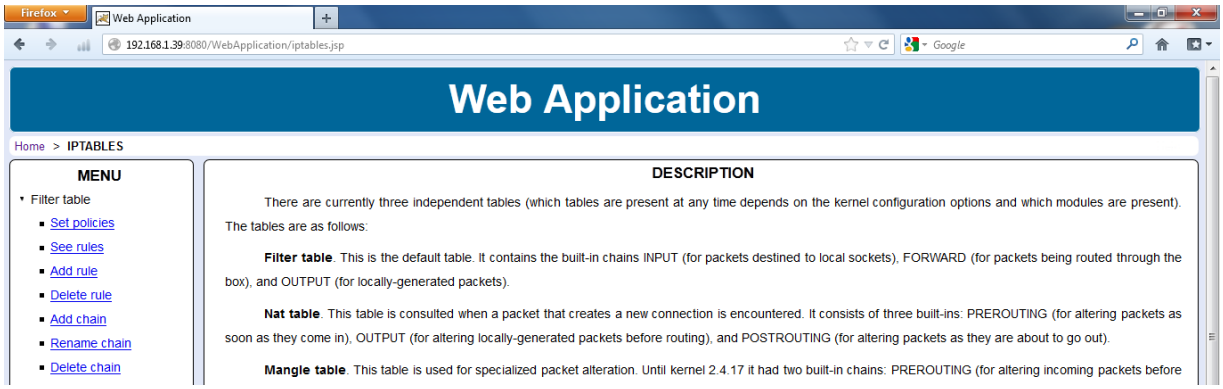

Figura A.15.: Acciones que pueden ser llevadas a cabo sobre la tabla filter de la herramienta IPtables.

Al seleccionar la acción "Set policies" se accede a la pantalla que se muestra en la Figura (A.16). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está gestionando.

2. Entrada para la actualización de los parámetros de la tabla que se está actualizando.

3. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

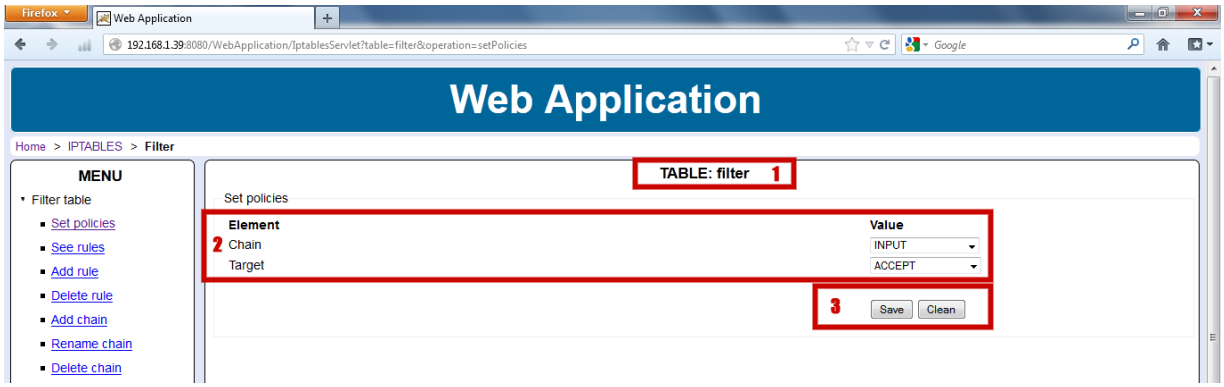

Figura A.16.: Pantalla para la gestión de las políticas por defecto de cada una de las cadenas de una tabla de IPtables.

Visualización de las reglas

Al seleccionar la acción "See rules" que aparece en el menú que se muestra tras hacer click sobre "Filter table", tal y como se presentaba en la Figura (A.15), se accede a la pantalla que se muestra en la Figura (A.17). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está visualizando.

2. Reglas, organizadas en función de la cadena a la que pertenecen y en el orden en el que son aplicadas, de la tabla que se está visualizando.

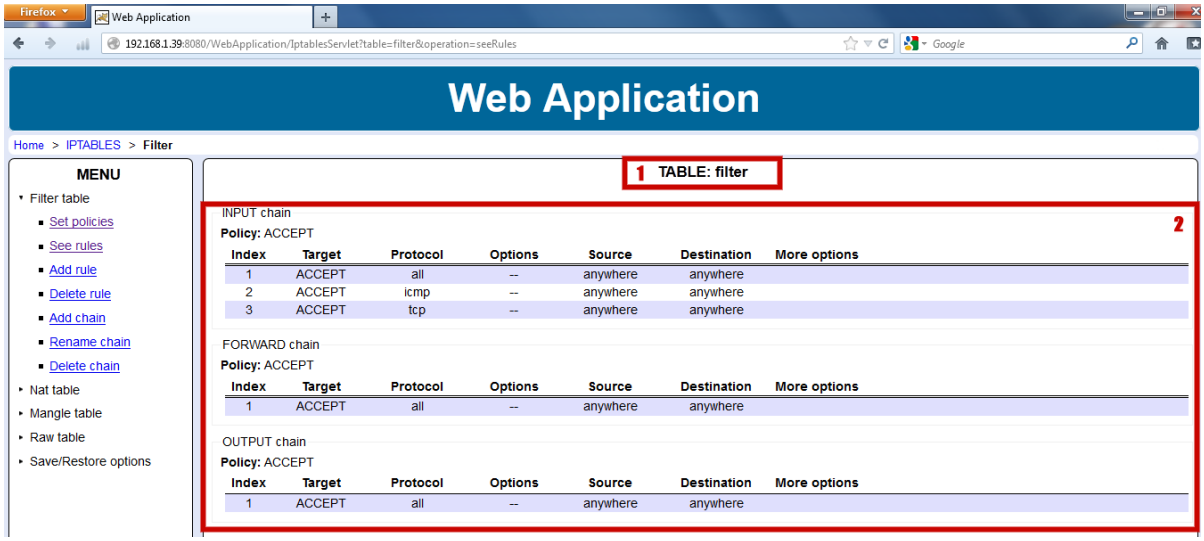

Figura A.17.: Pantalla para la visualización de las reglas de una tabla de IPtables.

Creación de una regla

Al seleccionar la acción "Add rule" que aparece en el menú que se muestra tras hacer click sobre "Filter table", tal y como se presentaba en la Figura (A.15), se accede a la pantalla que se muestra en la Figura (A.18). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está gestionando.

2. Nombre de los parámetros que pueden ser especificados en la regla que se va a crear.

3. Entrada para la actualización de los parámetros de la tabla que se está actualizando.

4. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

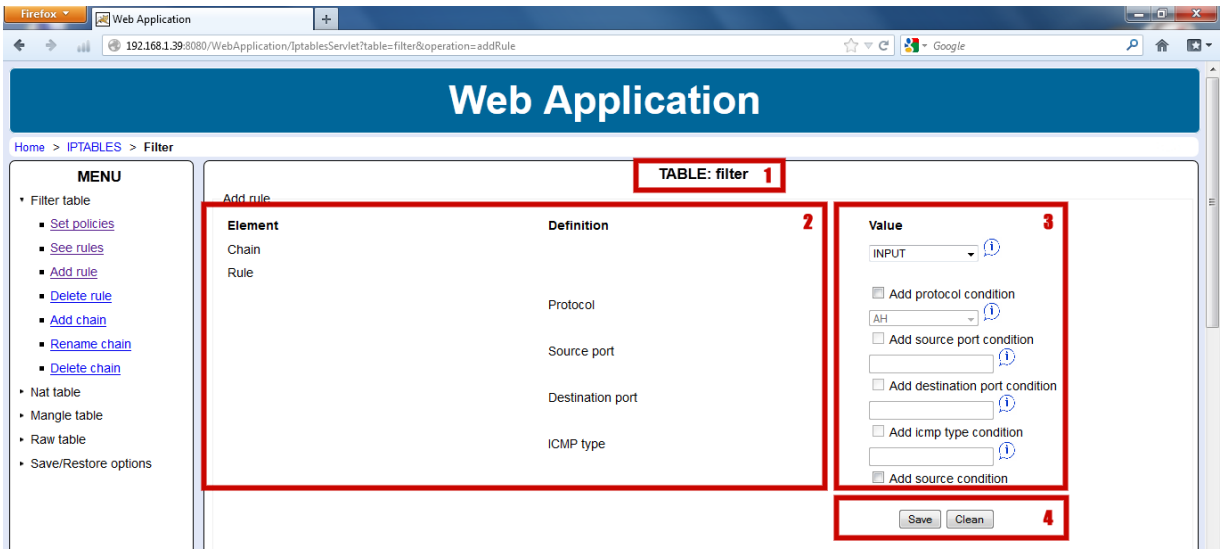

Figura A.18.: Pantalla para la creación de una regla en una tabla de IPtables.

Eliminación de una regla

Al seleccionar la acción "Delete rule" que aparece en el menú que se muestra tras hacer click sobre "Filter table", tal y como se presentaba en la Figura (A.15), se accede a la pantalla que se muestra en la Figura (A.19). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está gestionando.

2. Entrada para la actualización de los parámetros de la tabla que se está actualizando.

3. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

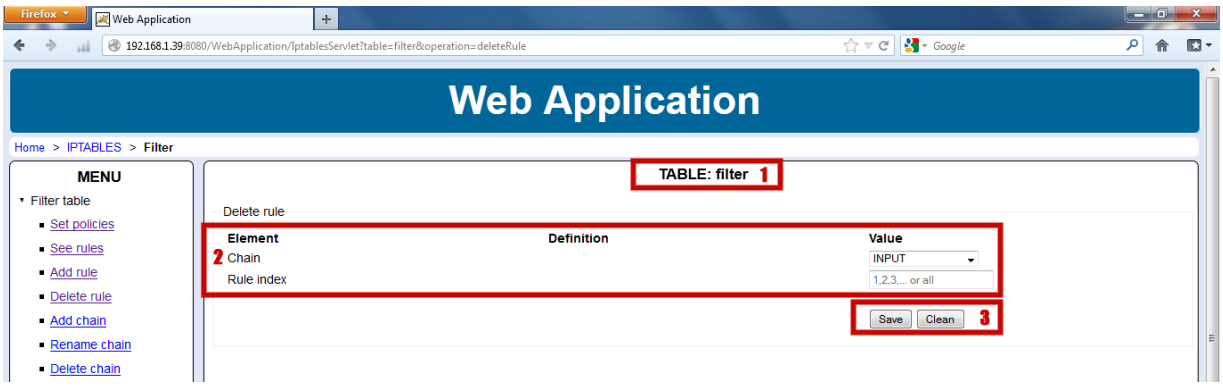

Figura A.19.: Pantalla para la eliminación de reglas de una tabla de IPtables.

Creación de una cadena

Al seleccionar la acción "Add chain" que aparece en el menú que se muestra tras hacer click sobre "Filter table", tal y como se presentaba en la Figura (A.15), se accede a la pantalla que se muestra en la Figura (A.20). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está gestionando.

2. Entrada para la actualización de los parámetros de la tabla que se está actualizando.

3. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

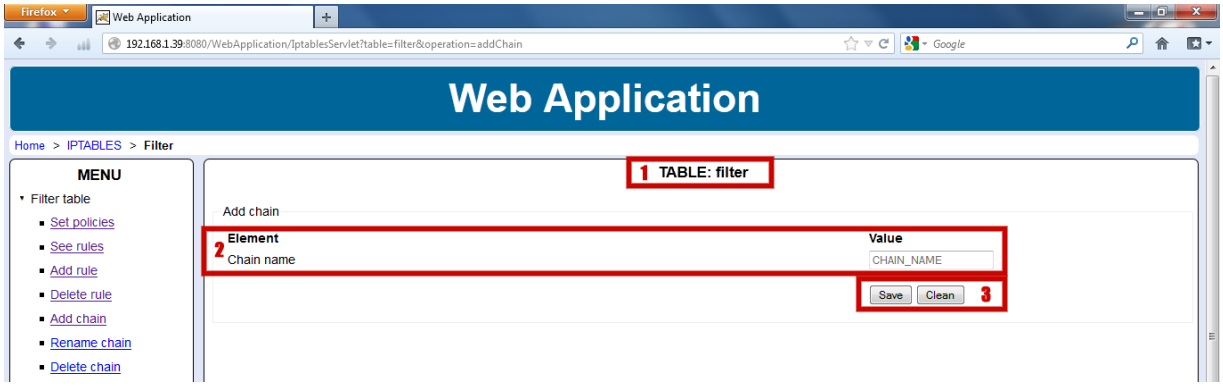

Figura A.20.: Pantalla para la creación de una cadena en una tabla de IPtables.

Renombrado de una cadena

Al seleccionar la acción "Rename chain" que aparece en el menú que se muestra tras hacer click sobre "Filter table", tal y como se presentaba en la Figura (A.15), se accede a la pantalla que se muestra en la Figura (A.21). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está gestionando.

2. Entrada para la actualización de los parámetros de la tabla que se está actualizando.

3. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

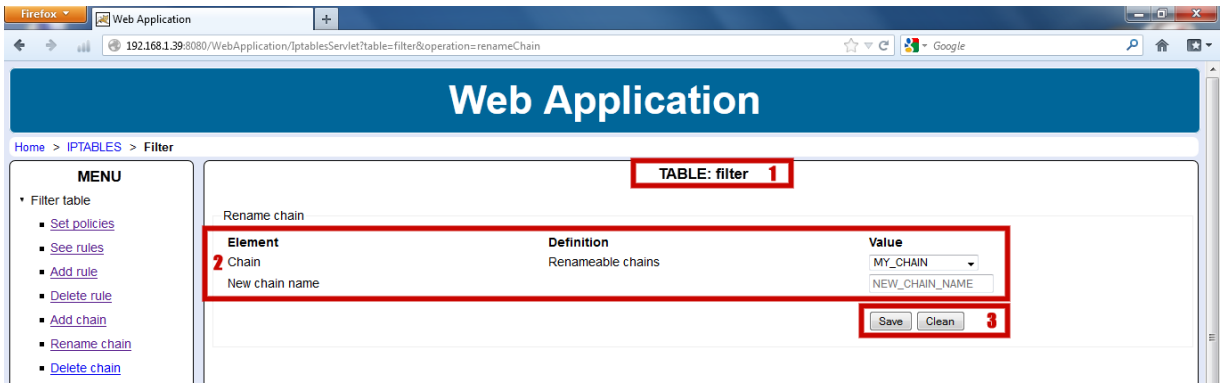

Figura A.21.: Pantalla para el renombrado de una cadena en una tabla de IPtables.

Eliminación de una cadena

Al seleccionar la acción "Delete chain" que aparece en el menú que se muestra tras hacer click sobre "Filter table", tal y como se presentaba en la Figura (A.15), se accede a la pantalla que se muestra en la Figura (A.22). En esta pantalla diferenciamos cada uno de los elementos enmarcados en la figura mencionada anteriormente:

1. Nombre de la tabla que se está gestionando.

2. Entrada para la actualización de los parámetros de la tabla que se está actualizando.

3. Botones para hacer permanentes los cambios y para limpiar los datos especificados en todas las entradas.

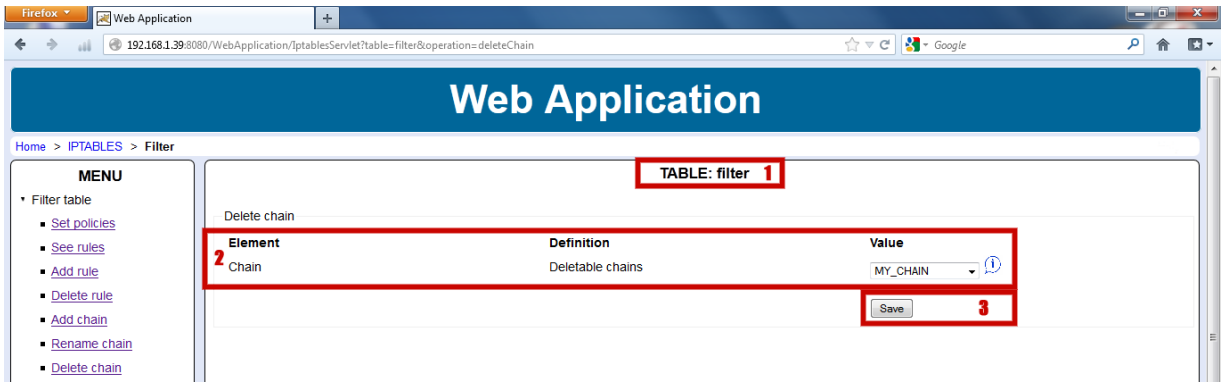

Figura A.22.: Pantalla para la eliminación de una cadena en una tabla de IPtables.

Guardado y restauración del estado de la aplicación IPtables

Al realizar cualquier cambio en el estado de la aplicación, la opción "Save/Restore options" del menú de IPtables se volverá de color rojo con el fin de indicar al usuario que ha aplicado modificaciones en el estado de la misma que no ha guardado[2]. Esto se muestra en la Figura (A.23).

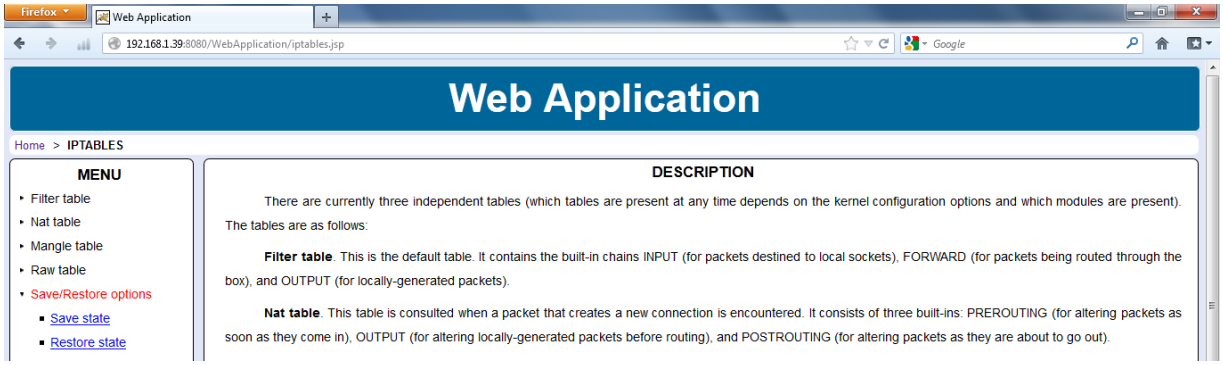

Figura A.23.: Pantalla principal de la gestión de IPtables.

Al seleccionar la acción "Save state" se accede a la pantalla que se muestra en la Figura (A.24). Para guardar el estado actual de la herramienta IPtables, basta con hacer click sobre el enlace "click here".

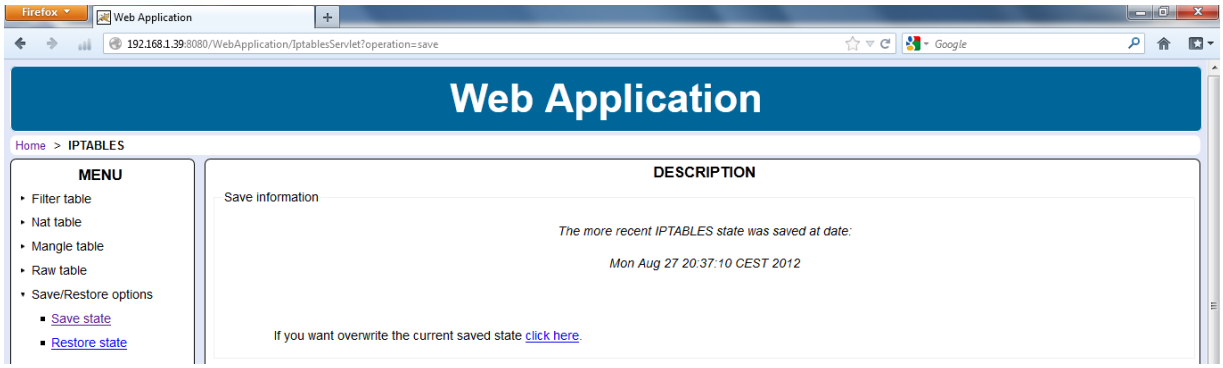

Figura A.24.: Pantalla para guardar el estado de IPtables.

Por otra parte, al seleccionar la acción "Restore state", siempre y cuando se halla guardado un estado de la herramienta IPtables anteriormente, se accede a la pantalla que se muestra en la Figura (A.25). Para restaurar el estado previamente guardado, basta con hacer click sobre el enlace "click here".

[2]Esta opción volverá a su color original tras haber guardado el estado de IPtables.

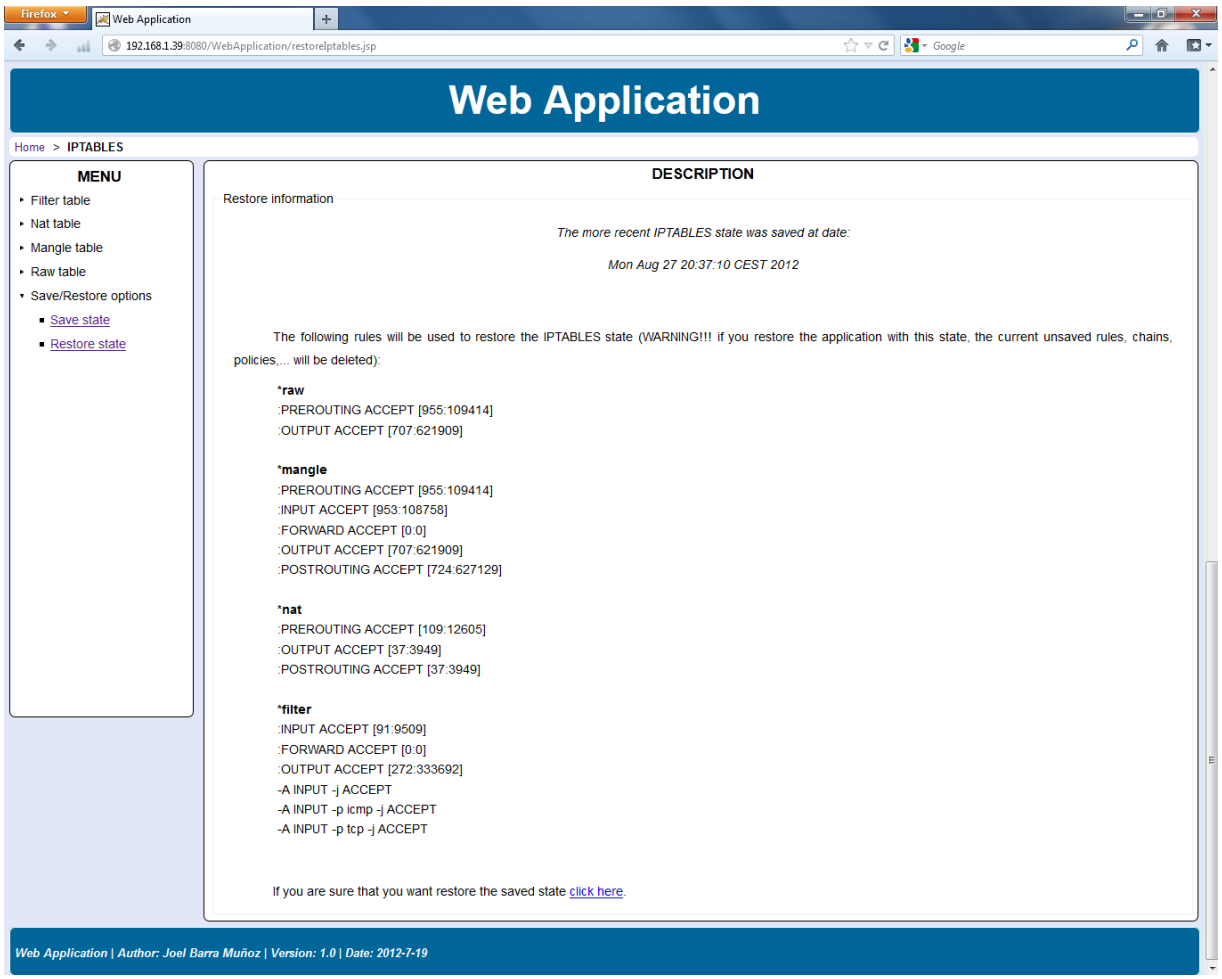

Figura A.25.: Pantalla para restaurar un estado de IPtables.